Proceedings

TWENTY-SIXTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

September 28-30, 1988

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

FEB 07 1989

NOTICE: **Return or renew all Library Materials! The *Minimum* Fee for each Lost Book is $50.00.**

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.
To renew call Telephone Center, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

MAR 20 1989
REC'D FEB 21 '90
JAN 19 REC'D
MAR 31 1989
MAY 17 REC'D
DUE: 5/31/89
JUN 9 REC'D
JUL 7 REC'D
INTERLIBRARY LOAN
OCT 20 REC'D
FEB 13 1990
JAN 16 REC'D
FEB 1994

L161—O-1096

PROCEEDINGS

TWENTY-SIXTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

VOLUME II

Mark W. Spong
Bruce Hajek
Conference Co-Chairmen

Conference held
September 28-30, 1988
Allerton House
Monticello, Illinois

Sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

Digitized by the Internet Archive
in 2026 with funding from
University of Illinois Urbana-Champaign

https://archive.org/details/proceedings1988alle_1

621.3811
Al5
1988:2

Engin

FOREWORD

The Twenty-Sixth Annual Allerton Conference on Communication, Control, and Computing was held at Allerton House, Monticello, Illinois on September 28-30, 1988. The conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. These *Proceedings* contain the papers presented at the Conference.

We are grateful to Professor Jack Keil Wolf of the University of California at San Diego for his plenary lecture "New Results for Magnetic and Optical Recording Channels." We would like to thank all of the session chairpersons and the authors for their contributions.

We appreciate the support of Professor M.E. Van Valkenburg, founder of the series of Allerton Conferences, Professor T.N. Trick, Head of the Department of Electrical and Computer Engineering, and Professor W.K. Jenkins, Director of the Coordinated Science Laboratory. We would also like our colleagues Professors J.A. Abraham, K. S. Arun, P. Banerjee, M.T. Basar, Y. Bresler, D.J. Brown, W.K. Fuchs, T.S. Huang, R. Iyer, P.V. Kokotovic, P.R. Kumar, H. Lee, M.C. Loui, J.V. Medanic, D.C. Munson, J.H. Patel, W.R. Perkins, K.R. Poolla, H.V. Poor, F.P. Preparata, M.B. Pursley, V. Ramachandran, D.V. Sarwate, and B.W. Wah for their generous assistance in reviewing papers (over 360 were submitted) and organizing the accepted papers into sessions. A special thanks goes to Professor G. Manacher at The University of Illinois at Chicago for his assistance to the program committee.

We are grateful to Mr. Steven Williams for his handling of the business affairs of the Conference, to Mr. Robert Ebeling and his assistants for their timely printing of large numbers of programs and registration packets, to Jennifer Eickman and her assistants at the Allerton House who cheerfully accommodated our group, and especially to Ms. Dixie Murphy, Ms. Janet Reese, Ms. Lila Rhoades, and Ms. Sharon Teuben-Rowe for their capable assistance in handling the Conference announcement, registration, program preparation, and editing and sales of these *Proceedings*. These individuals make the Conference co-chairmen look good by doing most of the work. Finally, we would like to thank Ms. Christina Kamra of John Wiley for sponsoring a coffee break and book display.

M.W. Spong

B. Hajek

TABLE OF CONTENTS

FOREWORD i

I-A: ESTIMATION AND DETECTION

Chairman: J. Sacks
University of Illinois at Urbana-Champaign

AN OPTIMUM SEQUENCE ESTIMATION APPROACH IN THE ABSENCE OF SYNCHRONIZATION 1
C. N. Georghiades

COMPUTATION OF THE MINIMUM-ASN BOUNDARIES FOR THE MODIFIED SPRT: PERFORMANCE ANALYSIS AND COMPARISONS 11
O. Snorrason and F. D. Garber

A PARAMETRIC FORM FOR SIEVE-CONSTRAINED MAXIMUM-LIKELIHOOD DENSITY ESTIMATION FROM NOISY DATA 13
T. J. Schulz and D. L. Snyder

ASYMPTOTICALLY EFFICIENT ESTIMATION OF PRIOR PROBABILITIES IN MULTICLASS FINITE MIXTURES 15
G. R. Dattatreya and L. N. Kanal

ROBUST LOCALLY-OPTIMAL DETECTION OF DEPENDENT SIGNALS IN NON-GAUSSIAN NOISE 25
P. Willett and P. Srivastava

CRAMER-RAO BOUNDS FOR CONSTRAINED SPECTRUM ESTIMATION WITH APPLICATION TO A PROBLEM IN RADAR IMAGING 27
J. A. O'Sullivan, P. Moulin and D. L. Snyder

ROBUSTNESS IN TERMS OF A J-DIVERGENCE SIGNAL-TO-NOISE RATIO 35
D. Warren

RESISTANCE AND ROBUSTNESS FOR FINITE MEMORY DETECTORS 37
K. S. Vastola

DECISION FUSION IN DISTRIBUTED DETECTION WITH UNCERTAINTIES 39
E. Drakopoulos and C. C. Lee

I-B: COMPUTATION GEOMETRY

Chairman: Franco Preparata
University of Illinois at Urbana-Champaign

THE NUMBER OF MAXIMAL EDGE SEQUENCES AS A CONVEX POLYTOPE 49
C. Schevon and J. O'Rouke

TWO TECHNIQUES FOR CREATING EXTRA COMPONENTS TO OBTAIN LOWER BOUNDS 58
P. Ramanan

A SEARCH PROBLEM IN PLANE GEOMETRY 63
K. Sugihara, I. Suzuki and M. Yamashita

GEOMETRIC SELECTION BY PARAMETRIC SEARCH 73
J. S. Salowe

AN OPTIMAL ALGORITHM FOR CONSTRAINED DELAUNAY TRIANGULATION 83
D.-M. Jung

LOWER BOUNDS FOR SOME CONSTRAINED VORONOI DIAGRAMS 85
C. A. Wang

A UNIFIED APPROACH TO FINDING ENCLOSING AND ENCLOSED TRIANGLES 87
D. M. Mount and S. Chandran

COVERING ORTHOGONAL POLYGONS WITH SQUARES 97
L. J. Aupperle, H. E. Conn, J. M. Keil and J. O'Rourke

AN EFFICIENT ON-LINE LINEAR CHEBYSHEV APPROXIMATION ALGORITHM FOR A FINITE PLANAR POINT SET 107
B. K. Bhattacharya

I-C: MULTIPROCESSOR ARCHITECTURES AND ALGORITHMS

Chairman: Wen-Mei Hwu
University of Illinois at Urbana-Champaign

DESIGN AND ANALYSIS OF A HIERARCHICAL SNOOPING CACHE COHERENCE SYSTEM 109
M. Azimi and C. Erickson

A PERFORMANCE STUDY FOR PARALLEL COMPUTATION ON LOOSELY-COUPLED MULTIPROCESSOR SYSTEMS 119
L. Liu, J.-K. Peir, A. Bross and D. Shen

A MAPPING ALGORITHM FOR MULTIPROCESSOR ARCHITECTURES 129
S. Yalamanchili and D. T. Lee

SYSTEMS WITH NEARLY ZERO DISTRIBUTED SIMULATION OVERHEAD 139
D. Kumar

DISTRIBUTED COMPUTATION ON CONCURRENT PROCESSORS TIME SHARED MULTICOMPUTING 149
V. K. Madisetti and D. G. Messerschmitt

I-D: LINEAR SYSTEMS

Chairman: Juraj V. Mendaic
University of Illinois at Urbana-Champaign

NEW RESULTS ON EIGENVALUE CONCEPTS FOR LINEAR TIME VARYING DYNAMICAL SYSTEMS 159
J. Zhu and C. D. Johnson

IMPLUSE RESPONSES AND KERNEL REPRESENTATIONS 169
D. Ball and I. W. Sandberg

CLASSICAL AND EXTENDED TRANSMISSION ZEROS: MODEL MATCHING RELATIONSHIPS 177
M. K. Sain, B. F. Wyman and J. L. Peczkowski

AN EXTENSION OF KHARITONOV'S THEOREM FOR A CLASS OF MULTILINEAR COEFFICIENT PERTURBATIONS 187
B. R. Barmish and Z. Shi

INVARIANT SUBZEROS ARE NOT INVARIANT 189
C. B. Schrader and M. K. Sain

ON STABLE SOLUTIONS OF THE ONE AND TWO SIDED MODEL MATCHING PROBLEMS 191
Z. Gao and P. J. Antsaklis

SYSTEMS IDENTIFICATION AND MODEL REDUCTION OF LINEAR CONTINUOUS SYSTEMS FROM IMPULSE RESPONSE DATA 199
P. Agathoklis and V. Sreeram

NEAR-OPTIMAL MODEL REDUCTION IN BALANCED AND MODAL COORDINATES 209
W. Gawronski and J.-N. Juang

I-E: DIGITAL SIGNAL PROCESSING

Chairman: K. S. Arun
University of Illinois at Urbana-Champaign

A PARAMETRIC METHOD FOR DETERMINING THE NUMBER OF SIGNALS 219
Q. Wu and D. R. Fuhrmann

AN EXTENDED OBE ALGORITHM FOR ARMA PARAMETER ESTIMATION 229
A. K. Rao, Y.-F. Huang and S. Dasgupta

NONLINEAR ITERATIVE TECHNIQUES FOR IMAGE RESTORATION FROM INCOMPLETE SPECTRAL DATA 239
K. D. Sanders and A. C. Bovik

EIGENANALYSIS BASED BISPECTRUM ESTIMATION 249
D. L. Sherman and M. D. Zoltowski

OPTIMAL DISCRIMINATION OF MULTIDIMENSIONAL AUTOREGRESSIVE PROCESSES 251
M. M. Milosavljevic

II-A: PARALLEL ALGORITHMS

Chairman: Vijaya Ramachandran
University of Illinois at Urbana-Champaign

EFFICIENT PARALLEL ALGORITHMS FOR STRING EDITING AND RELATED PROBLEMS 253
A. Apostolico, M. J. Atallah, L. L. Larmore and H. S. McFaddin

FAST PARALLEL ALGORITHMS FOR BIPARTITE PERMUTATION GRAPHS 264
L. Chen and Y. Yesha

PT-OPTIMAL ALGORITHMS FOR INTERVAL GRAPHS 274
A. Moitra and R. Johnson

FAST PARALLEL ALGORITHMS FOR TREE DECOMPOSING AND PARSING PARTIAL k-TREES 283
N. Chandrasekharan and S. T. Hedetniemi

AN OPTIMAL PARALLEL ALGORITHM FOR PREEMPTIVE JOB SCHEDULING THAT MINIMIZES MAXIMUM LATENESS 293
S. H. Rodger

FINDING A MINIMAL COVER FOR BINARY IMAGES: AN OPTIMAL PARALLEL ALGORITHM 303
D. Moitra

II-B: NONLINEAR SYSTEMS

Chairman: Ren Jeng Su
University of Colorado at Boulder

STABILITY ANALYSIS OF ALMOST SINUSOIDAL PERIODIC OSCILLATIONS IN NONLINEAR CONTROL SYSTEMS SUBJECTED TO NONCONSTANT PERIODIC INPUT 313
G. S. Krenz and R. K. Miller

PULSE-WIDTH-MODULATION CONTROLLER DESIGN FOR NONLINEAR DELAY DIFFERENTIAL SYSTEMS 321
H. Sira-Ramirez

DESIGN OF NONCHAOTIC FEEDBACK SYSTEMS 331
V. Raman and R.-W. Liu

LIAPUNOV FUNCTIONS FOR CRITICAL NONLINEAR SYSTEMS 338
J.-H. Fu and E. H. Abed

ON STABILIZATION WITH A PRESCRIBED REGION OF ASYMPTOTIC STABILITY 348
L. Saydy, E. H. Abed and A. L. Tits

APPLICATION OF HAMILTONIAN SYSTEM THEORY TO THREE-MACHINE POWER SYSTEM TRANSIENT ANALYSIS 356
T. Guo, N. Narasimhamurthi and N. H. McClamroch

IDENTIFICATION OF A CLASS OF PARALLEL SYSTEMS 366
M. Pawlak

II-C: FAULT TOLERANCE AND DIAGNOSIS

Chairman: Kent Fuchs
University of Illinois at Urbana-Champaign

USING REPLICATION TO IMPLEMENT RELIABLE STORAGE 368
B. Bhargava and S. Browne

CORRESPONDENT COMPUTING BASED SOFTWARE FAULT TOLERANCE 378
P.-N. Lee, A. Tamboli and J. Blankenship

IMPROVED UPPER BOUNDS FOR ALGORITHM-BASED FAULT TOLERANCE 388
D. J. Rosenkrantz and S. S. Ravi

ADAPTIVE DIAGNOSIS FOR PROBABILISTICALLY DIAGNOSABLE SYSTEMS 398
F.-H. W. Shih and K. Nakajima

PROPERTIES OF c-CORRECTABILITY IN SELF-DIAGNOSING SYSTEMS 408
J. T. Butler and Y. Rosen

II-D: ADAPTIVE AND ARRAY SIGNAL PROCESSING

Chairman: W. K. Jenkins
University of Illinois at Urbana-Champaign

AVERAGING THEORY FOR SIGN-SIGN LMS 417
B. D. O. Anderson, I. M. Y. Mareels, W. A. Sethares and C. R. Johnson

ON MEAN SQUARE CONVERGENCE OF ADAPTIVE BEAM-FORMING ALGORITHMS 427
G. Yin

AN EFFICIENT ALGORITHM FOR TRACKING MULTIPLE MOVING TARGETS 437
C. R. Sastry, E. W. Kamen and M. Simaan

INSUFFICIENT ORDER ADAPTIVE IIR FILTERS WITH WHITE INPUTS 447
M. Nayeri

THE PERFORMANCE CHARACTERISTICS OF THE ORDER RECURSIVE ADAPTIVE LATTICE EQUALIZER 457
B. M. Sallam and S. B. Kesler

ADAPTIVE AVERAGE FILTERS FOR IMAGE PROCESSING 459
B. Chang and M. Kanefsky

II-E: INFORMATION THEORY AND CODING

Chairman: D. L. Snyder
Washington University

ASYMPTOTIC BOUNDS FOR FOCUSED ERROR CONTROL CODES 461
T. Fuja and C. Heegard

THE ENUMERATION OF (D,K) SEQUENCES 471
K. Forsberg and I. Blake

FOURIER TRANSFORM CODING FOR PEAK LIMITED CHANNELS 473
E. Feig

IMPROVING GENERALIZED MINIMUM DISTANCE DECODING BY COMPARISON WITH ERRORS-ONLY DECODING 480
D. Taipale

A UNIVERSAL OUTER CODE DECODING TECHNIQUE 482
J. J. Metzner

AN IMPROVED DECODING ALGORITHM FOR CERTAIN ELLIPTIC CODES 492
C.-M. Liu and P. V. Kumar

OPTICAL ORTHOGONAL CODES - NEW BOUNDS AND AN OPTIMAL CONSTRUCTION 494
H. Chung and P. V. Kumar

MAXIMUM MUTUAL INFORMATION DECODING FOR ARBITRARILY VARYING MULTIPLE-ACCESS CHANNELS 499
J. A. Gubner

III-A: VARIALBE STRUCTURE CONTROL AND ROBOTICS

Chairman: Ray DeCarlo
Purdue University

BINARY CONTROL SYSTEMS DESIGN 501
R. T. Yanushevsky

ANALYSIS AND SYNTHESIS OF A CLASS OF NEURAL NETWORKS: VARIABLE STRUCTURE SYSTEMS WITH INFINITE GAINS 508
J.-H. Li, A. M. Michel and W. Porod

ON THE ESTIMATION OF SLIDING DOMAINS AND STABILITY REGIONS OF VARIABLE STRUCTURE CONTROL SYSTEMS 518
S. M. Madani-Esfahani, S. Hui and S. H. Zak

THE ROLE OF SLIDING MODE IN CONTACT FORCE CONTROL OF ROBOT MANIPULATORS 528
K. D. Young

ON CONTINUOUS APPROXIMATION TO VARIABLE STRUCTURE CONTROL 535
F. Esfandiari and H. K. Khalil

STABILITY OF A CLASS OF LINEAR MECHANICAL SYSTEMS WITH LINEAR HOLONOMIC CONSTRAINTS 537
D. Wang and N. H. McClamroch

TIP POSITION ADAPTIVE CONTROL OF A FLEXIBLE MANIPULATOR 547
T.-C. Yang, J. C. S. Yang and P. Kudva

III-B: STOCHASTIC METHODS

Chairman: W. A. Massey
AT&T Bell Laboratories

COMPENSATORS AND DERIVATIVE ESTIMATION FOR QUEUEING SYSTEMS 549
M. A. Zazanis

MONTE CARLO SIMULATION AND LARGE DEVIATIONS THEORY: THE ASYMPTOTIC OPTIMALITY OF THE EXPONENTIAL SHIFT 556
J. S. Sadowsky, J. A. Bucklew and P. Ney

CUSTOMER ROUTING TO DIFFERENT SERVERS WITH COMPLETE INFORMATION 566
Z. Rosberg and P. Kermani

AVERAGE COST SEMI-MARKOV DECISION PROCESSES AND THE CONTROL OF QUEUEING SYSTEMS 569
L. I. Sennott

QUICKER CONVERGENCE FOR OPTIMAL STOPPING PROBLEMS THROUGH PROBABILISTIC INTERPRETATIONS OF CLASSICAL NUMERICAL METHODS 571
A. G. Greenberg and R. J. Vanderbei

ENTROPY RATE OF STATIONARY RANDOM FIELDS ON BINARY TREE 580
T. Berger and Z. Ye

MATRIX QUADRATIC EQUATIONS AND QUASI-BIRTH-AND-DEATH MODELS OF MULTIPLE ACCESS NETWORKS 589
G. R. Murthy and E. J. Coyle

ON FUSION IN ESTIMATION THEORY 599
E. B. Hall, A. E. Wessel and G. L. Wise

III-C: IMAGE PROCESSING AND ANALYSIS

Chairman: Yoram Bresler
University of Illinois at Urbana-Champaign

IMPROVING COVER TYPE IDENTIFICATION IN SPECKLED SAR IMAGES BY PREFILTERING AND SEQUENTIAL CLASSIFICATION 609
Q. Lin and J. Allebach

ESTIMATING CONVEX SHAPES FROM SUPPORT LINE MEASUREMENTS USING PRIOR GEOEMTRIC INFORMATION 619
J. L. Prince and A. S. Willsky

A TWO-DIMENSIONAL AUTOREGRESSIVE MODELING TECHNIQUE FOR ANALYSIS AND CLASSIFICATION OF PLANAR SHAPES 629
M. Das, M. J. Paulik and N. K. Loh

UNCERTAINTY AND SIMILARITY MEASURES IN PATTERN RECOGNITION 639
J. Dj. Golic

A MODEL AND SYNTHESIS OF TEXTURE ON A 3-D SURFACE 647
Y. Choe and R. L. Kashyap

EDGE DETECTION BY SIMULATED ANNEALING 657
H. L. Tan and E. J. Delp

STATISTICAL APPROACH TO NONDIFFRACTION COMPUTED TOMOGRAPHY IMAGING AND ITS APPLICATION IN IMAGE ANALYSIS 659
T. Lei and W. Sewchand

SIMULTANEOUS IDENTIFICATION AND RESTORATION OF IMAGES USING MAXIMUM LIKELIHOOD ESTIMATION AND THE EM ALGORITHM 661
K. T. Lay and A. K. Katsaggelos

III-D: DISTRIBUTED COMPUTATION AND COMPLEXITY THEORY

Chairman: M. C. Loui
University of Illinois at Urbana-Champaign

EFFICIENT AGREEMENT USING FAULT DIAGNOSIS 663
B. A. Coan

ON EFFICIENTLY COMPUTING FUNCTIONS OF DISTRIBUTED INFORMATION 673
T. V. Lakshman and V. K. Wei

ON FINDING STRONGLY CONNECTED COMPONENTS DISTRIBUTIVELY 683
K. Brahmadathan and S. Venkatesan

IMPROVING PETERSON'S CONSTRUCTION OF l-WRITER n-READER MULTIVALUED ATOMIC REGISTER 693
K. Vidyasankar

AN ASYNCHRONOUS ALL PAIRS SHORTEST PATH ALGORITHM FOR SHARED MEMORY MULTIPROCESSORS 701
A. Uresin and M. Dubois

TEST CASE CONSTRUCTION FOR NP-HARD PROBLEMS 711
L. A. Sanchis

RELATIVE COMPLEXITY OF EVALUATING AND CONSTRUCTING THE OPTIMUM OF MAXIMIZATION PROBLEMS 721
P. Crescenzi and R. Silvestri

III-E: TESTING AND CONCURRENT ERROR DETECTION

Chairman: J. Patel
University of Illinois at Urbana-Champaign

TESTING OF CMOS CIRCUITS AT FUNCTIONAL LEVEL 731
L. Zhang, L. Tong and R.-W. Liu

ON THE COMPLEXITY OF COMPUTING TESTS FOR CMOS GATES 740
S. Chakravarty

A HIERARCHICAL SEQUENTIAL TEST GENERATOR 750
W.-T. Cheng and J.-Y. Jou

ON THE DESIGN OF VLSI CMOS PLAs FOR BIST IMPLEMENTATION 758
R. Dandapani, R. K. Gulati, D. K. Goel and K. R. Callahan

SFS/SSC DOMINO-CMOS IMPLEMENTATIONS OF TSC CIRCUITS 768
N. K. Jha

FAULT COLLAPSING FOR CONCURRENT ERROR DETECTION ANALYSIS 778
L. L. Kinney and T. Ramakrishnan

IV-A: COMBINATORIAL ALGORITHMS

Chairman: C. L. Liu
University of Illinois at Urbana-Champaign

SEPARATORS, GRAPH HOMOMORPHISMS AND CHROMATIC POLYNOMIALS 788
H. B. Hunt, III, S. S. Ravi and R. E. Stearns

ALGORITHMS FOR BISECTING PLANAR GRAPHS 798
T. N. Bui and A. Peck

A BETTER CLIQUE-PARTITIONING ALGORITHM 808
J. Bhasker and T. Samad

ADAPTIVE AND OPTIMAL RANDOMIZATION OF ALGORITHMS IN P: A CASE STUDY 810
W. H. Janko, A. Geyer-Schulz and A. Taudes

THE SET UNION PROBLEM WITH DYNAMIC WEIGHTED BACKTRACKING 819
G. Gambosi, G. F. Italiano and M. Talamo

FINDING NORMAL FORMS, CLIQUES, HAMILTONIAN PATHS AND HAMILTONIAN CIRCUITS IN A SET OF PROPER CIRCULAR ARCS IN OPTIMUM TIME AND SPACE 832
G. K. Manacher and T. A. Mankus

LINEAR TIME ALGORITHMS ON CIRCULAR-ARC GRAPHS 842
W.-L. Hsu and K. Tsai

IV-B: CONTROL SYSTEMS DESIGN

Chairman: S. H. Zak
Purdue University

NEW ESTIMATES OF ASYMPTOTIC TRAJECTORY BOUNDS FOR DIGITAL FILTERS: IMPROVED RESULTS 852
J. A. Farrell and A. N. Michel

A GENERALIZED LEAD-LAG COMPENSATION TECHNIQUE 862
N. M. Arbouz and R. Su

A POLE-PLACEMENT PROCEDURE VIA H^{∞}-OPTIMIZATION 872
A. E. Frazho and D. K. Le

ROBUSTNESS OF FEEDBACK SYSTEMS WITH SEVERAL SOURCES OF MODELLING ERROR 881
J. S. Freudenberg and J. Chen

REPRESENTATION OF UNCERTAIN ACTUATOR DYNAMICS AND DECENTRALIZED ROBUST CONTROLLER DESIGN 883
A. Iftar and U. Ozguner

TIME DOMAIN CONTROL DESIGN FOR ROBUST STABILITY OF DISCRETE-TIME LINEAR REGULATORS 893
S. R. Kolla and J. B. Farison

SPECTRAL APPROXIMATION OF ALMOST BLOCK-TRIANGULAR OPERATORS AND HANKEL-NORM APPROXIMATION OF TWO FREQUENCY SCALE TRANSFER MATRICES 903
H. Oloomi and M. E. Sawan

BLOCK TRIANGULARIZATION OF DISCRETIZED MULTIPARAMETER SINGULARLY PERTURBED SYSTEMS 905
G. L. Syrmos and E. H. Abed

IV-C: STOCHASTIC AND ADAPTIVE CONTROL

Chairman: Han-Fu Chen
University of Illinois at Urbana-Champaign

AN ADAPTIVE CONTROLLER TO ACHIEVE AN ARBITRARILY GOOD TRANSIENT AND STEADY-STATE RESPONSE 907
D. E. Miller and E. J. Davison

A ROBUST ADAPTIVE CONTROL FOR LINEAR PLANTS IN THE PRESENCE OF UNMODELED DYNAMICS 917
S. Bai and F. M. A. Salam

LIAPUNOV BASED MODEL REFERENCE LEARNING CONTROL 927
M. Phan and R. W. Longman

ON LINEAR SYSTEMS DRIVEN BY SELF-EXCITED POINT PROCESSES 937
M. A. Ingram and A. H. Haddad

ADAPTIVE STABILIZATION OF MULTI-MACHINE POWER SYSTEMS 939
H. Yousef and M. Simaan

THE EQUIVALENCE OF STABILIZABILITY CONDITIONS FOR STOCHASTIC SYSTEMS 948
E. Yaz

DRIFT TRANSFORMATIONS USING FEEDBACK CONTROLS AND APPLICATIONS TO SMALL-NOISE PROBLEMS 953
E. Kappos

IV-D: COMMUNICATION NETWORKS

Chairman: L. I. Sennott
Illinois State University

A SPLITTING ALGORITHM FOR COMMUNICATION WITH REAL-TIME DELIVERY CONSTRAINTS 955
D. C. Marinescu

ON THE STABILITY CONDITION OF A PRECEDENCE-BASED QUEUEING DISCIPLINE 965
F. Baccelli and Z. Liu

SIMPLE, EFFICIENT COMPUTATION OF THE GENERALIZED THROUGHPUT OF JACKSON NETWORKS 967
A. G. Greenberg and W. A. Massey

A LOAD-CONTROLLED TOKEN RING PROTOCOL 969
Y. W. Chiang and C. C. Lee

USE OF FLOW CONTROL IN A FINITE USER RANDOM ACCESS SYSTEM.................................. 979
A. B. Behroozi-Toosi and R. R. Rao

PARALLEL TIME COMPLEXITY OF ROUTING IN PERMUTATION NETWORKS.. 981
D. M. Koppelman and A. Y. Oruc

LIMITED WAITING FOR VALUABLE RESOURCES: AN ADAPTIVE CONTROL STRATEGY FOR CIRCUIT-SWITCHED NETWORKS.. 991
G. Gopal and A. Weinrib

VOICE TRANSMISSION IN PACKET SWITCHING NETWORKS: A MODEL FOR QUEUEING ANALYSIS.. 1001
F. M. Brochin and J. B. Thomas

IV-E: ALGORITHMS AND ARCHITECTURES FOR SIGNAL AND IMAGE PROCESSING

Chairman: Hua Lee
University of Illinois at Urbana-Champaign

A NEW APPROACH TO SOLVE THE SEQUENCE-LENGTH CONSTRAINT PROBLEM IN CIRCULAR CONVOLUTION USING NUMBER THEORETIC TRANSFORM.. 1005
H. Lu and S. C. Lee

A FAMILY OF OPTIMAL ARCHITECTURES FOR MULTIDIMENSIONAL TRANSFORMS.. 1015
C. Chakrabarti and J. JaJa

PARALLEL ARCHITECTURE FOR BOUNDARY FOLLOWING OF REGIONS OF AN IMAGE STORED AS LINEAR QUADTREE.. 1025
R. B. Panwar and S. K. Nandy

OPTIMAL VISIBILITY ALGORIITHMS FOR BINARY IMAGES ON THE HYPERCUBE.. 1035
F. Dehne, Q. T. Pham and I. Stojmenovic

A VLSI PARALLEL PROCESSOR FOR BICUBIC SURFACE PATCH IMAGE GENERATION.. 1037
P. C. Chao and M.-Y. Chern

GRAMMARS AND BAYES' PRIORS FOR HIERARCHICAL IMAGE PROGESSING ON MASSIVELY PARALLEL PROCESSORS... 1040
B. Roysam and M. I. Miller

A ROTATION BASED ALGORITHM FOR FAST LEAST SQUARE DECONVOLUTION.. 1042
Y. H. Hu and P. H. Milenkovic

RESIDUE COMPLEX MULTIPLIERS.. 1044
A. Skavantzos and T. Stouraitis

EFFICIENT IMPLEMENTATION OF CHIRP Z-TRANSFORM USING CORDIC ALGORITHM.. 1046
Y. H. Hu and S. Naganathan

PLENARY LECTURE

NEW RESULTS FOR MAGNETIC AND OPTICAL RECORDING CHANNELS 1048
J. K. Wolf
University of California at San Diego

V-A: TRELLIS CODING

Chairman: W. E. Stark
University of Michigan

ON THE PATH WEIGHT ENUMORATORS OF CONVOLUTIONAL CODES 1049
C. C. Chao and R. J. McEliece

AN ARBITRARILY TIGHT UNION BOUND FOR CONVOLUTIONAL CODES THROUGH BIFURCATION 1059
M. V. Burnashev, A. S. Arora and D. L. Cohn

CONSTRUCTION OF GOOD DISTANCE PROFILE TRELLIS CODES FOR SEQUENTIAL DECODING 1061
S. S. Malladi, D. J. Costello, Jr. and H. C. Ferreira

THE APPLICATION OF TRELLIS CODING TECHNIQUES TO DIRECT-SEQUENCE SPREAD-SPECTRUM MULTIPLE ACCESS COMMUNICATIONS 1070
W. E. Stark and B. D. Woerner

UNIFORM DISTANCE AND ERROR PROBABILITY PROPERTIES OF TCM SCHEME 1080
E. Biglieri and P. J. McLane

CONSTRUCTION OF TRELLIS-DECODABLE ERROR-CORRCTING LINE CODES 1082
M. A. Herro, V. Telang and A. R. Calderbank

V-B: PARALLEL COMPUTATION

Chairman: L. B. Pitt
University of Illinois at Urbana-Champaign

DETERMINISTIC SIMULATIONS OF PRAMS ON BOUNDED DEGREE NETWORKS 1084
K. T. Herley and G. Bilardi

MAPPING PYRAMID ALGORITHMS INTO HUYPERCUBES 1094
T.-H. Lai and W. White

OPTIMIZATION OF COMPUTATION TIME FOR SYSTOLIC ARRAYS 1104
Y. Wong and J.-M. Dolosme

TOPOLOGICAL NUMBERING OF FEATURES ON A MESH 1114
M. J. Atallah, S. E. Hambrusch and L. E. TeWinkel

SOLUTION OF LINEAR EQUATIONS ON SHUFFLE-EXCHANGE AND MODIFIED SHUFFLE EXCHANGE NETWORKS 1116
R. B. Panwar and L. M. Patnaik

V-C: VLSI LAYOUT

Chairman: S. S. Ravi
State University of New York at Albany

EMBEDDING RECTANGULAR GRIDS INTO SQUARE GRIDS WITH DILATION TWO 1126
R. Melhem and G.-Y. Hwang

GENERAL CIRCULAR PERMULATION LAYOUT 1136
R. D. Lou and M. Sarrafzadeh

THE ASSIGNMENT HEURISTIC FOR CROSSING REDUCTION IN BIPARTITE GRAPHS 1138
T. Catarci

V-D: DISCRETE EVENT SYSTEMS AND NEURAL NETWORKS

Chairman: D. L. Elliott
National Science Foundation

A STATE-VARIABLE APPROACH TO THE MODELING AND CONTROL OF DISCRETE-EVENT SYSTEMS 1140
Y. Li and W. M. Wonham

MODELING OF INTERACTING CONTINUOUS TIME AND DISCRETE EVENT SYSTEMS: AN EXAMPLE 1150
P. Peleties and R. DeCarlo

BRANCHING TIME TEMPORAL LOGIC FOR DISCRETE EVENT SYSTEM ANALYSIS 1160
K. M. Passino and P. J. Antsaklis

A DECOMPOSITION APPROACH FOR SENSITIVITY ANALYSIS OF DISCRETE-EVENT SYSTEMS 1170
S. G. Strickland and C. G. Cassandras

NEURAL COMPUTING FOR INFORMATION EXTRACTION IN CONTROL SYSTEMS 1172
K. M. Passino, M. A. Sartori and P. J. Antsaklis

MORE ON PARETO OPTIMAL FLOW CONTROL 1181
C. Douligeris and R. Mazumdar

INDEX OF AUTHORS

IMPROVING COVER TYPE IDENTIFICATION IN SPECKLED SAR IMAGES BY PREFILTERING AND SEQUENTIAL CLASSIFICATION *

QIAN LIN and JAN ALLEBACH
Laboratory for Applications of Remote Sensing and
School of Electrical Engineering
Purdue University, West Lafayette, IN 47907

ABSTRACT: Synthetic aperture radar utilizes coherent microwaves to produce images of the earth's surface. Due to the interference of coherent wavelets, the images appear speckled. This reduces the performance of per-pel classifiers. One way to increase the performance is to filter the image first, then classify the filtered image. For this purpose, we investigate several novel filters that have been reported in the literature. These are the geometric filter, adaptive LMMSE filter, and linear approximation filter. For comparison, we also consider conventional mean and median filters. In our experiments, we find that the mean filter with seven iterations gives the best result. The overall performance increased from 65.2% to 88.9%. We also assess the capability of these filters to preserve edges in the original image. We find that the geometric and median filters are the best in preserving edges, and that the linear approximation and adaptive LMMSE filters are the best in discriminating roads. Prefiltering the image effectively provides contextual information to the per-pel classifier. An alternate approach is to directly design a contextual classifier. We propose a new contextual classifier based on sequential decision theory. With this classifier, we find that the overall performance increases to 89.5%.

1. INTRODUCTION

Classification of cover type from remotely sensed images of the earth's surface provides an important means of gathering information about natural resources. Such images have traditionally been acquired with passive multispectral scanners operating in the visible and infrared; but more recently, active synthetic aperture radar (SAR) operating in the microwave region has also been used. In this case, images may be acquired at different incidence angles and with different transmitter/receiver polarizations to provide diversity corresponding to the different spectral bands of a multispectral scanner. Compared to multispectral scanners, SAR possesses the advantage of being able to operate day or night independent of weather or cloud conditions. However, SAR is a coherent system; and images acquired with it are degraded by speckle noise due to interference of waves reflected from multiple scatterers within each resolution cell. Speckle noise severely degrades the performance of per-pel classifiers which have been used successfully with multispectral images. In addition, speckle noise impedes identification of image detail, which is essential for correlating image information with that gathered from other sources including ground truth. Such information is, in fact, needed to identify the test samples on which the classifier design is based.

One approach to this problem is to filter the images to reduce speckle prior to classification. Mueller [1] has shown that conventional linear and median filters can improve classifier performance. However, these filters tend to degrade image sharpness to varying degrees. Recently, several novel filters have been developed with the objective of reducing speckle while maintaining image sharpness. These include the geometric filter [2] [3], an adaptive linear, minimum mean squared error (LMMSE) filter [4] [5], and a linear approximation approach in which the SAR image is expressed in terms of a basis set consisting of multispectral images [6]. The first objective of this paper is to evaluate the effect of these filters on classifier performance.

---

* Research supported by NASA under Contract No. NAGW-925.

Besides classification performance, another criteria for comparing filters is visual assessment. We define measures for edge spread and road contrast, and use them to evaluate the visual quality of SAR images processed with the various filters.

The filter-then-classify approach may be regarded as a rather indirect means of providing a per-pel classifier with contextual information. A second approach is to develop explicitly contextual classifiers that operate directly on the speckled image. Classifiers based on combining similar neighbors [7] and on compound decision theory [8] are examples. Image segmentation techniques [9] are also applicable. The second objective of this paper is to develop and evaluate a contextual classifier based on sequential decision theory [10].

In Sec. 2, we briefly describe the three new filters to be evaluated in the filter-then-classify approach. In Sec. 3, we also briefly describe the conventional per-pel Bayes classifier. We then present our new sequential classifier. In Sec. 4, we define the measures for assessment of visual quality. Finally, Sec. 5 contains the results from an experimental comparison of the various filter and classifier combinations.

2. FILTERS FOR SPECKLE REDUCTION

The geometric filter [2] [3] is essentially a morphological operator [11]. It may be explained as follows: Suppose a discrete two-dimensional gray level image is converted to a discrete three-dimensional volume with the additional z axis representing the gray value. Points below the gray level surface have value 1; and those above it have value 0. The volume is partitioned into parallel slices perpendicular to the x-y plane to obtain a set of binary images. Each binary image is filtered by running four 3×3 templates over it. Wherever a template matches, the center pixel is complemented; otherwise it remains the same. The filtered image is then complemented, and processed again in the same manner with four additional templates. Complementing the output of this step yields the final filtered output for the slice. The above procedure is applied sequentially to sets of slices taken at four different angles with respect to the x-axis. This constitutes one iteration. The final result is obtained by converting the binary images back to the original image form. It is important to note that this can always be done since the overall filtering operation obeys the stacking property [12].

The adaptive filter [4] [5] is based on a model in which the noise is multiplicative, and spatially uncorrelated with a unit-mean Gaussian distribution. Thus the (i,j)-th pixel $z_{i,j}$ in the SAR image is related to the terrain reflectance $x_{i,j}$ and speckle noise $v_{i,j}$ according to $z_{i,j} = x_{i,j}\, v_{i,j}$. Suppressing the subscripts (i,j), the LMMSE estimate of the terrain reflectance may be expressed as

$$\hat{x} = \bar{x} + k(z - \bar{x}),$$

where

$$k = \frac{\sigma_x^2}{\bar{x}^2\sigma_v^2 + \sigma_x^2},$$

and

$$\sigma_x^2 = \frac{\sigma_z^2 + \bar{z}^2}{\sigma_v^2 + \bar{v}^2} - \bar{x}^2.$$

The parameters σ_z^2 and $\bar{z}$ are calculated as sample averages within a neighborhood of the (i,j)-th pixel. Since the SAR image is an amplitude image preprocessed with a four-look average on intensity basis, $\sigma_v = 0.2536$ [5]. In addition, $\bar{v} = 1$; and this implies that $\bar{x} = \bar{z}$. In a flat region, $\sigma_x^2 \simeq 0$, and $\hat{x} \simeq \bar{x}$, which is the linear average of pixels in the neighborhood. In a high contrast region or edge area, σ_x^2 is large. In this case, $\hat{x} \simeq z$, and the value of the pixel is not changed.

Use of the approximation method [6] depends on the availability of multispectral images covering the same terrain as the SAR images, and which are registered with them. The multispectral images are highly correlated with the SAR images, but are not speckled. Thus, they comprise a good basis set for a smooth, linear approximation to the SAR images.

To determine the smoothed image value $\hat{x}$ at a given pixel, we express the SAR image data $\vec{z}$ within a w-point window in terms of the p reference images $\vec{y}_i$, i=1, ..., p, as

$$\begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ z_w \end{bmatrix} = \begin{bmatrix} 1 & y_{11} & \cdot & \cdot & y_{1p} \\ 1 & y_{21} & \cdot & \cdot & y_{2p} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & y_{w1} & \cdot & \cdot & y_{wp} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \cdot \\ \cdot \\ b_p \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ \cdot \\ \cdot \\ e_p \end{bmatrix}$$

or in vector-matrix notation,

$$\vec{z} = Y\vec{b} + \vec{e},$$

where $\vec{e}$ is the approximation error. The coefficients $\vec{b}$ are chosen to give a least squares fit:

$$\vec{b} = (Y^T Y)^{-1}(Y^T\vec{z}).$$

The smoothed SAR image value is then given by

$$\hat{x} = [1 \ \ y_{k1}, \ldots, y_{kp}]\ \vec{b}.$$

where k is the index of the center pixel in the window.

3. CLASSIFICATION

With per-pel classifiers, each image pixel is classified independently of the others in the image. Let $\vec{x}$ be a 3-tuple of the SAR image values for the three incidence angles at a particular pixel; and suppose that it belongs to one of M classes. Assuming that the distribution of $\vec{x}$ conditioned on it belonging to the i-th class is multivariate Gaussian with mean $\vec{\mu}_i$ and covariance Σ_i, the discriminant function for a Bayes classifier is given by

$$g_i(\vec{x}) = \ln(p_i) - \frac{1}{2}\ln(\left|\Sigma_i\right|) - \frac{1}{2}(\vec{x} - \vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x} - \vec{\mu}_i).$$

Here p_i is the prior probability of the i-th class. The pixel is placed in that class i^* for which

$$g_{i^*}(\vec{x}) \geqq g_i(\vec{x}), \quad i = 1,\ldots,M.$$

Since the signal to noise ratio of SAR images is very low, the per-pel classification rate is low compared to that of images obtained from noncoherent imaging systems. Filtering the speckled images reduces the variance of the data, making the gray value vector for each pixel more representative of the class to which it belongs. This results in an improvement in classifier performance.

Our contextual classifier examines the values of pixels in a neighborhood around the pixel in question to aid in its classification. The sequence in which these pixels are evaluated is indicated in Fig. 1. The pixel to be classified is evaluated first. At the n-th step in the algorithm, we attempt to classify this pixel based on its value $\vec{x}_1$ and that of its n-1 neighbors $\vec{x}_2, \ldots, \vec{x}_n$. If no decision can be made, we add the pixel $\vec{x}_{n+1}$ to the set and repeat the process. To reduce the chance of running across a boundary between two cover types, we limit the neighborhood to a 5×5 window.

We first describe the classification algorithm when there are only two classes H_0 and H_1. We then discuss the extension to the case of $M > 2$ classes. Assuming neighboring pixels are independent and belong to the same class, we have, at the n-th stage of the algorithm, n i.i.d. vector observations $\vec{x}_1, \ldots, \vec{x}_n$. To test the hypothesis H_1 versus H_0, we compute the log-likelihood ratio

15	16	10	17	18
14	9	5	6	19
13	4	1	2	11
25	8	3	7	20
24	23	12	22	21

Fig. 1. Data sequence used in the sequential classifier.

$$s_n = \ln\left(\frac{f(\vec{x}_1, \ldots, \vec{x}_n \mid H_1)}{f(\vec{x}_1, \ldots, \vec{x}_n \mid H_0)}\right) = \sum_{k=1}^{n} \ln\left(\frac{f(\vec{x}_k \mid H_1)}{f(\vec{x}_k \mid H_0)}\right),$$

and compare it with two fixed thresholds $a < 0 < b$. If $s_n \leqq a$, we choose H_0; and if $s_n \geqq b$, we choose H_1. In this case, the algorithm terminates. Otherwise, $a < s_n < b$; so we get another data sample $\vec{x}_{n+1}$, and repeat the process.

To bound the error probabilities by α_0 and α_1

$$P\ (\text{choosing } H_1 \mid H_0) \leqq \alpha_0,$$

$$P\ (\text{choosing } H_0 \mid H_1) \leqq \alpha_1,$$

we let

$$a = \ln\left(\frac{\alpha_1}{1-\alpha_0}\right),$$

$$b = \ln\left(\frac{1-\alpha_1}{\alpha_0}\right).$$

For M classes, we calculate the log-likelihood functions of the data sequence for each class i

$$l_{i,n} = \ln(\, f(\vec{x}_1, \ldots, \vec{x}_n \mid H_i)),$$

and then calculate the log-likelihood ratio $s_n = l_{i1,n} - l_{i2,n}$ for the largest two of the log-likelihood functions. If we can make a decision based on s_n, we terminate. Otherwise, we add a sample, recalculate the likelihood functions for each class, and again attempt to make a decision as above. If a decision cannot be made with all 25 samples in the window, we choose the class with the largest likelihood function.

4. EDGE RENDITION AND ROAD DISCRIMINATION

To quantitatively describe the sharpness of edges, we define the edge spread S to be the distance between the two pixels which are the 90% transition points shown in Fig. 2. In digital imagery, it is easier to detect edges horizontally and vertically than at an arbitrary angle. Suppose we scan horizontally, and detect an edge. We calculate the spread S of the edge as if the edge were vertical. If the edge is at an angle α with the vertical direction, as shown in Fig. 3, we calculate a least squares estimate of the slope and scale the spread of the edge accordingly.

More precisely, let the x coordinates range over the integers from 0 to N-1, and the y coordinates be the distances to the edge, as shown in Fig. 3. The edge can be represented by the linear equation

$$y = mx + b,$$

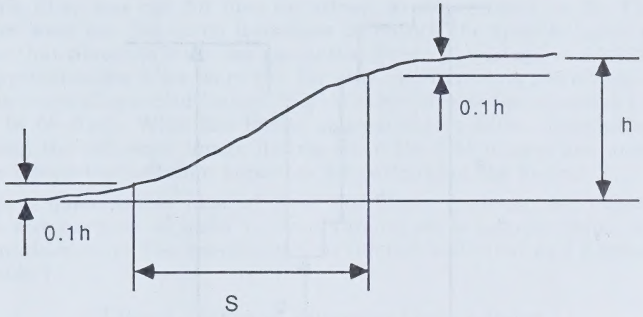


Fig. 2. Definition of edge spread.

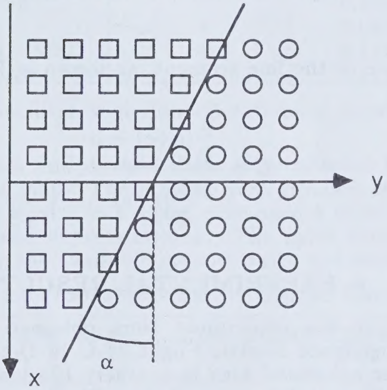


Fig. 3. Edge slope estimation.

where m is the slope we are trying to estimate. Let

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{N-1} & 1 \end{bmatrix}, \quad X = \begin{bmatrix} m \\ b \end{bmatrix}, \quad B = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_{N-1} \end{bmatrix};$$

then $AX \simeq B$. The least squares fit of the parameters m and b is given by

$$X = (A^TA)^{-1}A^TB.$$

Since the x_i are the integers from 0 to N-1, $(A^TA)^{-1}A^T$ can be precalculated. The corrected edge spread is then given by

$$S_\alpha = S\cos(\alpha) = \frac{S}{\sqrt{1+m^2}}.$$

In SAR images, pixels representing roads have a smaller gray value than surrounding pixels. Suppose we use an array to represent a line segment across a road. Let a and c be the gray values of the two end points of the line segment, while b is

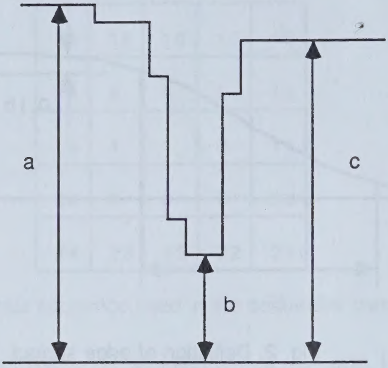


Fig. 4. Definition of road contrast.

the minimum gray value of the line segment, as shown in Fig. 4. We define the contrast of the road to be

$$C = \frac{\frac{1}{2}(a+c) - b}{\frac{1}{2}(a+c)}.$$

5. EXPERIMENTAL RESULTS

The images used in the experiment were obtained by the Shuttle Imaging Radar-B (SIR-B) during Space Shuttle Flight 41-G in October, 1984. Three sets of data were collected over a forested area in northern Florida on Oct. 9, 10, and 11, at incidence angles 58°, 45°, and 28°, respectively. The images were digitally processed by the Jet Propulsion Laboratory (JPL) in Pasadena, CA [13]. A set of Thematic Mapper (TM) images over the same area was obtained on Oct. 12, 1984; and they were registered with the SIR-B images by JPL. The images we used here have 512×512 pixels. Each pixel covers a 28.5×28.5 m^2 area on the ground. The SAR images with incidence angles 28°, 45°, and 58° are referred to as bands 1, 2, and 3, respectively. The TM images have seven bands, which are used as the reference images in the linear approximation filter.

By calculating the statistics of several sample fields, it was found that there is not a great difference in correlation between different cover types, which means that the "texture" we see in the image does not contain much information about cover type. In fact, if we subtract the local mean and scale by the local standard deviation making the transformation

$$\tilde{x}_{ij} = (x_{ij} - \bar{x}) / \sigma_x + 128$$

for every pixel in the image, the interior region of different ground cover types cannot be distinguished in the resulting image, at least not visually, although the roads and sharp edges are still present. On the other hand, if we only subtract the local mean, or only scale by the local standard deviation, then there still exist visible differences between ground cover types. This suggests that most of the information about ground cover type is contained in the local mean and variance. The speckle index $\sigma_z/\bar{z} \simeq 0.26$ in most of the fields. This is consistent with the expected value of 0.2536 discussed in Sec. 2.

The images were filtered with the geometric filter, the adaptive LMMSE filter, the linear approximation filter, and conventional 3×3 mean and 3×3 median filters.

The geometric filter was run for four iterations, as is suggested in [3]. The mean and median filters were run for seven iterations to reduce the speckle index to about the same level as that obtained with the geometric filter. The adaptive LMMSE filter and the linear approximation filter were run for only one iteration, since they are based on models of the original speckled image. The window size of the adaptive LMMSE filter was chosen to be 5×5. With the linear approximation filter, singularity is possible when inverting the reference image matrix since the TM images are so smooth. This problem was circumvented when necessary by perturbing the matrix **Y**.

To gain a quantitative idea of how the filters perform, we concentrate on a 32×32 pixel water region of band 1. Since this region is homogeneous, the variations are due to speckles only. The speckle indices for the unfiltered and filtered images are shown in Table 1.

Table 1. Effect of Filters on Speckle Index

Filter Type	No. Iterations	Speckle Index
original image	0	0.2627
3×3 mean	7	0.1025
3×3 median	7	0.1000
geometric	4	0.0832
adaptive LMMSE	1	0.1376
linear approx.	1	0.1717

We see that the geometric filter with four iterations is most effective in reducing speckle noise.

Eighteen training fields and 42 test fields were selected based on local lumber companies' inventories and maps. These fields were grouped into six different classes: water, clear cut with trees aged 1 to 6 years, pine aged 6 to 36 years, pine older than 36 years, mixed swamp, and cypress swamp. The prior probabilities of each class were obtained by analyzing the available ground truth, and were set as in Table 2.

Table 2. Prior Probabilities for Six Ground Cover Types

Cover Type	Prior Probability
Water	0.03
Clear cut and pine (1 - 3 yrs.)	0.16
Pine (6 - 36 yrs.)	0.35
Pine (> 36 yrs.)	0.20
Mixed swamp	0.20
Cypress swamp	0.06

Classification was performed on the original image, on images filtered with the geometric filter, the adaptive LMMSE filter, and the linear approximation filter, as well as the 3×3 mean and median filters. The classifier's performance was estimated using the holdout method [14]. The results are shown in Fig. 5. We see that using filtered images greatly improves the classification rate. The mean filter with seven iterations gave the best results.

For the sequential classifier, we chose the error probabilities to be $\alpha_0 = \alpha_1 = 0.01$. We did the sequential probability ratio test on the original and filtered images. The classification rate is also shown in Fig. 5. We can see that the classification rate increased in all cases by varying degrees. The largest increase occurred when sequential classification was done on the raw data. Note that the classification rate did not reach the theoretical error probability. This is mainly because the theoretical value is for a binary hypothesis test. Other reasons include truncation of the data sequence and statistical inhomogeneity of the images.

To gain a qualitative idea of the characteristics of the different filters, we show in Fig. 6, portions of the original image, the images filtered by the geometric filter, the mean filter, and the median filter. These portions each contain 256×256 pixels. We see that the geometric filter preserved edges very well, while effectively reducing fluctuations due to the speckle noise. Compared with the geometric filter, the mean

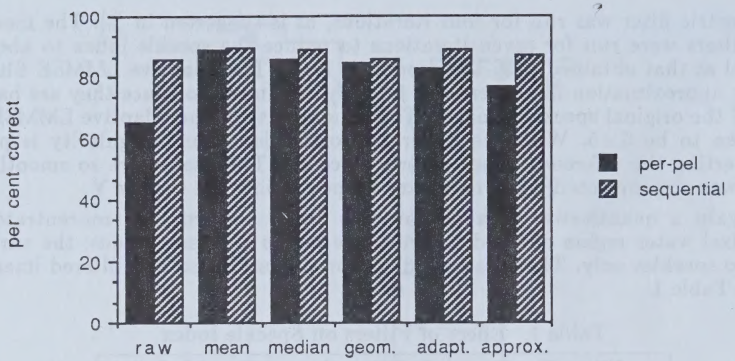


Fig. 5. Performance of per-pel and sequential classifiers.

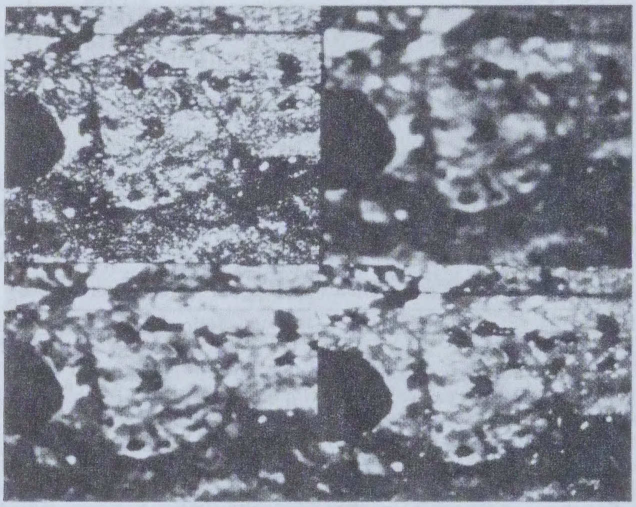


Fig. 6. Portion of original image and images filtered with the mean, median and geometric filters (left to right and top to bottom).

filter smeared edges. Edge rendition with the median filter is similar to that of the geometric filter, although the median filter obliterated some distinct features present in the original image, which were preserved by the geometric filter. Figure 7 shows portions of the original image, the image filtered by the linear approximation filter, the TM image (band 5) and the image filtered by the adaptive LMMSE filter. The linear approximation filter has the special characteristic of enhancing the edges which are blurred and hardly distinguishable in SIR-B images, but which are very clear in the TM images, since it uses the TM images to approximate the SIR-B images. As expected, the adaptive filter passed edges in the original image with little change.

We calculated the edge spread and road contrast for the various filtered images. To detect edges, we used the boundary information of a classified image, because the original SIR-B image are severely speckled. For the same reason, we used a TM image to detect roads. The result is shown in Table 3.

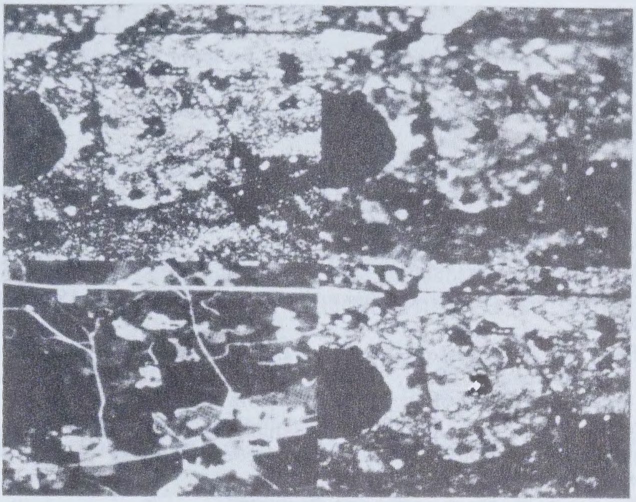


Fig. 7. Portion of original image, image filtered with the adaptive LMMSE filter, TM image (band 5), and image filtered with the linear approximation filter (left to right and top to bottom).

Table 3. Edge Spread and Road Contrast Measures

Filter Type	No. Iterations	Edge Spread S	Road Contrast C
original image	0	2.794	0.289
3×3 mean	7	5.317	0.077
3×3 median	7	3.767	0.105
geometric	4	3.500	0.103
adaptive LMMSE	1	3.473	0.157
linear approx.	1	3.114	0.221

The linear approximation and adaptive LMMSE filters result in the smallest edge spread and the largest road contrast. With approximately the same speckle index, the geometric filter preserves edges and roads better than the mean and median filters.

6. CONCLUSION

Prefiltering the images improved the performance of the per-pel Bayes classifier by up to 36.3%. The mean filter gave the best classification rate, but was inferior to the other filters in edge rendition and road discrimination. The linear approximation and adaptive LMMSE filters yielded the best edge spread and road contrast. The contextual sequential classifier increased the classification rate by up to 37.3% compared to per-pel classification of the original image. With this classifier, performance did not vary much between the original image and images processed by the various filters.

ACKNOWLEDGMENT

The authors wish to thank Prof. R. M. Hoffer, Dr. D. F. Lozano, and Mr. K. Lee for their assistance in providing the SAR and TM images and the ground truth information, and for many helpful discussions. One of the authors (J. A.) also wishes to thank Prof. D. C. Munson, Jr. for pointing out the geometric filter.

REFERENCES

[1] P. W. Mueller, *Spatial Filtering of Shuttle Imaging Radar-B Data,* M.S. Thesis, Purdue Univ., 1987.

[2] T. R. Crimmins, "Geometric Filter for Speckle Reduction," *Appl. Opt.,* Vol. 24, pp. 1438-1443, May 1985.

[3] T. R. Crimmins, "Geometric Filter for Reducing Speckle," *Opt. Eng.,* Vol. 25, pp. 651-654, May 1986.

[4] J. S. Lee, "Speckle Suppression and Analysis for Synthetic Aperture Radar," *Opt. Eng.,* Vol. 25, pp. 636-643, May 1986.

[5] J. S. Lee, "Statistical Modelling and Suppression of Speckle in Synthetic Radar Images," *Proc. of IGARSS '87 Symposium,* Ann Arbor, MI, 18-21 May 1987, pp. 1331-1339.

[6] V. T. Tom, M. J. Carlotto, "Adaptive Least-Squares Technique for Multi-band Image Enhancement," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 704-707, March 1985.

[7] R. L. Kettig and D. A. Landgrebe, "Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects," *IEEE Trans. Geosci Electron.,* Vol. GE-14, pp. 19-26, Jan. 1976.

[8] P. H. Swain, S. B. Vardeman, and J. C. Tilton, "Contextual Classification of multispectral image data," *Pattern Recognition,* Vol. 13, No. 6, pp. 429-441, 1981.

[9] A. Rosenfeld and A. C. Kak, *Digital Picture Processing,* Vol. 2, 2nd Ed., New York: Academic Press, 1982.

[10] A. Wald, *Sequential Analysis,* New York: Dover, 1947.

[11] J. Serra, *Image Analysis and Mathematical Morphology,* New York: Academic Press, 1982.

[12] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, "Stack Filters", *IEEE Trans. Acoust. Speech Signal Process.,* Vol. ASSP-34, pp. 898, August 1986.

[13] J. C. Curlander, "Performance of the SIR-B Digital Imaging Processing Subsystem," *IEEE Trans. Geosci. Remote Sensing,* Vol. GE-24, pp. 649-652, July 1986.

[14] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach,* Englewood Cliffs, NJ: Prentice Hall, 1982.

Estimating Convex Shapes from Support Line Measurements Using Prior Geometric Information[*]

Jerry L. Prince[†] and Alan S. Willsky
Laboratory for Information and Decision Systems
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge, MA 02139

September 26, 1988

Abstract

In this paper we present several algorithms for reconstructing 2-D convex sets given support line measurements for which the angles are known precisely but the lateral displacements are noisy. We extend the algorithms given in [5] by explicitly incorporating prior information about the shape of the objects to be reconstructed. In particular, the prior shape information is contained in a prior probability on support vectors, where a *support vector* is a vector formed from the lateral displacements of a particular set of support lines of an object. In order to relate the support vector prior to the expected shape of an object we develop a vector decomposition called the Size/Shape/Shift decomposition which helps to provide insight into the detailed geometric relationship between support vectors and 2-D convex objects. We then use the maximum a posteriori (MAP) criterion to determine the specific form of the support vector estimator. The computations involve a quadratic programming optimization stage, which is used to determine one component of the decomposition, and either a line search or conjugate gradient stage, which is used to determine the remaining components. The performance of the algorithms is demonstrated using simulated support line measurements of an ellipse.

1 Introduction

The problem which originally motivated this work is that of object reconstruction in computed tomography (CT) [1]. Specifically, in CT one makes measurements of the integrals of an object property (e.g., X-ray density) along various straight lines. The full set of line integrals for varying lateral displacements at a particular angle yields a projection. The usual goal of CT is to reconstruct an image of the object given a set of measured projections obtained from over a range of angles. We have found in other research [4], however, that in highly noisy and limited- or sparse-angle situations that it is beneficial to utilize

---

[*] This research was supported by the National Science Foundation under grant ECS-87-00903 and the U.S. Army Research Office under grant DAAL03-86-K-0171.

[†] The Analytic Sciences Corporation, Reading, MA 01867.

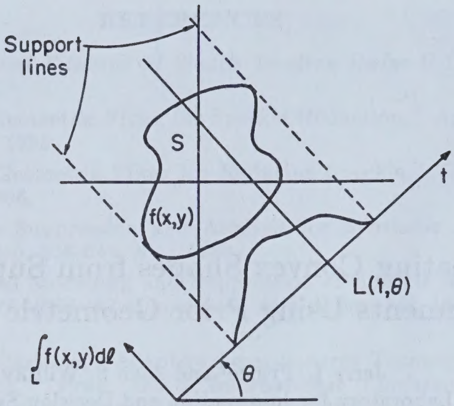


Figure 1: The geometry of computed tomography.

an estimate of the convex hull of the object in order to help reduce artifacts that would otherwise appear in the reconstructed imagery. This paper presents several methods to incorporate certain prior geometric information to aid in obtaining high quality convex hull estimates.

In the CT problem, an ideal projection contains information about the position of two lines — called *support lines* — which just graze the object in the plane (see Fig. 1). Given a set of such support lines, determined over many different angles, one may determine a convex polyhedron which circumscribes the object by intersecting all of the halfplanes defined by the measurements. In the limit as the angular spacing between projections goes to zero one may determine precisely the convex hull of the object. This type of sensor information and reconstruction problem is also of interest in several other applications including tactile sensing [7], silhouette imaging [9], robot vision [2], and chemical component analysis [3].

When the projections are noisy, such as in the case of low-dose CT, the estimates of the lateral positions of the support lines will also be noisy. In this case, the set of measured lines may be inconsistent — that is, taken together, there may be no set S which has all of the measured lines as support lines. In such cases, the consistency property of support lines developed in [5] together with a precise description of the noise properties may be used to make a maximum likelihood (ML) estimate of the true support vector. From this support vector estimate, an estimate of the convex hull of the object may be determined in a straightforward manner.

In this paper, we develop insights into the geometry of support lines and into the specification of prior probabilities on support lines that reflect particular prior information about the shape of the objects of interest. The major contributions are twofold: 1) the development of the Size/Shape/Shift (SSS) decomposition of support vectors, and 2) the

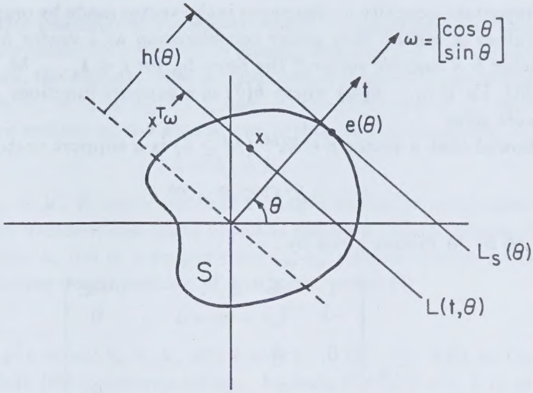


Figure 2: The geometry of support lines.

development of algorithms for convex shape estimation which utilize prior shape information. The emphasis in this paper is on circular and elliptical shapes, but the formulation is quite general and may be used to derive estimators for convex shapes described by other prior shape information.

2 Support Line Constraints

Fig. 2 shows what is meant by the support line $L_S(\theta)$ of a set S. It is the line orthogonal to the unit normal ω which just "grazes" S in the positive ω direction. The quantity $h(\theta)$ is the value of the largest possible projection of any point in S onto the ω-axis. One can see that S lies completely in a particular one of the two halfplanes determined by $L_S(\theta)$. We may now define the above quantities precisely. The *support line* at angle θ for the closed and bounded 2-D set S is given by

$$L_S(\theta) = \{x \in \mathbb{R}^2 \mid x^{\mathrm{T}}\omega = h(\theta)\} \tag{1}$$

where $\omega = [\cos\theta \;\; \sin\theta]^{\mathrm{T}}$ and

$$h(\theta) = \sup_{x \in S} x^{\mathrm{T}}\omega \;. \tag{2}$$

The function $h(\theta)$ is called the *support function* of the set S; for any particular value of θ we call $h(\theta)$ the *support value* at angle θ. In this paper we consider a finite number M of angles $\theta_i = 2\pi(i-1)/M$, $i = 1, \ldots, M$, spaced evenly over $[0, 2\pi)$, and associated sets of lines L_i, orthogonal to the corresponding unit vector $\omega_i = [\cos\theta_i \;\; \sin\theta_i]^{\mathrm{T}}$ and with lateral displacement h_i:

$$L_i = \{x \in \mathbb{R}^2 \mid x^{\mathrm{T}}\omega_i = h_i\} \;. \tag{3}$$

The most important quantity in this paper is the vector made by organizing the M lateral displacement values of the M lines under consideration as a vector $h = [h_1\ h_2\ \ldots\ h_M]^{\mathrm{T}}$. We call the vector h a ***support vector*** if the lines L_i, for $i = 1, \ldots, M$ are support lines for some set $S \in \mathbb{R}^2$, i.e. if $h_i = h(\theta_i)$ where $h(\theta)$ is a support function. In this case we refer to h_i as a ***support value***.

In [5] we showed that a vector $h \in \mathbb{R}^M$ $(M \geq 5)$ is a support vector if and only if

$$h^{\mathrm{T}} C \leq [0 \ldots 0] \tag{4}$$

where C is an M by M matrix given by

$$C = \begin{bmatrix} 1 & -k & 0 & & & -k \\ -k & 1 & -k & \ldots & \ldots & 0 \\ 0 & -k & 1 & & & \vdots \\ \vdots & 0 & -k & \ldots & \ldots & 0 \\ 0 & \vdots & & & & -k \\ -k & 0 & 0 & & & 1 \end{bmatrix} \tag{5}$$

and $k = 1/(2\cos(2\pi/M))$. It then follows that the convex polyhedral cone given by

$$\mathcal{C} = \{h \in \mathbb{R}^M \mid h^{\mathrm{T}} C \leq [0 \ldots 0]\} \tag{6}$$

consists of all M-dimensional support vectors. We call $\mathcal{C}$ the ***support cone***.

3 Geometry

Given a support vector h, there is, in general, an infinitely large family of sets which have h as their support vector. The largest of these sets, which is uniquely determined by h is the convex polygonal set S_B given by

$$S_B \;=\; \{x \in \mathbb{R}^2 \mid x^{\mathrm{T}}[\omega_1\, \omega_2 \ldots \omega_M] \leq [h_1\, h_2 \ldots h_M]\}\ . \tag{7}$$

We call S_B the ***basic object*** of support vector h. The shape of the basic object is, of course, determined by h, but h may be decomposed into several quantities which depend upon its position in the support cone, and have meaningful geometric interpretations with respect to the basic objects in the plane. We now briefly explore the Size/Shape/Shift (SSS) decomposition of a support vector.

It turns out that the matrix C is singular and a basis for the nullspace $\mathcal{N}$ is found to be [5]

$$\begin{aligned} n_1 &= \begin{bmatrix} 1 & \cos\theta_o & \cos 2\theta_o & \ldots & \cos(M-1)\theta_o \end{bmatrix}^{\mathrm{T}} \\ n_2 &= \begin{bmatrix} 0 & \sin\theta_o & \sin 2\theta_o & \ldots & \sin(M-1)\theta_o \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{8}$$

where $\theta_0 = 2\pi/M$. The geometrical consequence of C being singular is that the support cone $\mathcal{C}$ is not a proper cone; i.e., there is a linear subspace (of dimension 2) contained entirely in $\mathcal{C}$. Therefore, the support cone is composed of the Cartesian product of a proper cone, $\mathcal{C}_p = \{h \in \mathcal{C} \mid h^{\mathrm{T}} n_1 = 0,\ h^{\mathrm{T}} n_2 = 0\}$, and $\mathcal{N}$, the nullspace of C. Accordingly, any support vector may be written as the sum of two orthogonal components, h_p and h_n, as

$$h = h_p + h_n \tag{9}$$

where $h_p \in \mathcal{C}_p$ and $h_n \in \mathcal{N}$. It turns out that adding a nullspace component to a support vector yields a support vector whose basic object is simply a *shifted* version of the original.

Now since the vector h_p lies in a proper cone $\mathcal{C}_p$, h_p may be written in a type of radial form to yield the following decomposition of a support vector [4]

$$h = tq + h_n \tag{10}$$

where $t \geq 0$, $q \in \mathcal{C}_p$, $q^{\mathrm{T}} e = M$, $h_n \in \mathcal{N}$, and $e = [1\,1\ \ldots\ 1]^{\mathrm{T}}$. We refer to this relationship as the Size/Shape/Shift (SSS) decomposition. In fact, the quantity t is proportional to the circumference of the basic object S_B, which gives t the interpretation as the *size* of a basic object. It follows that q contains the information about the *shape* of the basic object. Therefore, given the SSS decomposition, we have a way to relate the t, q, and h_n components of a support vector to the size, shape, and position, respectively, of its basic object. We exploit this in the next section by designing prior probabilities which suit our expectations about the shape of the objects of interest.

One useful quantity which helps to characterize the shape of the basic object corresponding to h is the discrete radius of curvature r_i which we define to be $r_i = -(2/\theta_0 \tan\theta_0) h^{\mathrm{T}} c_i$, where c_i is the ith column of the matrix C in (4). In [5] we show this quantity to be analogous to the continuous radius of curvature of smooth closed convex curves. In the discrete case, r_i is proportional to the length of the face of the basic object whose unit outward normal is given by ω_i. Therefore, a small r_i corresponds to an object which has a sharp bend in its boundary at the ith face; a larger r_i corresponds to a bend which is not as sharp.

4 Scale-Invariant Algorithms

In this section we consider a class of prior probabilities on support vectors h which are *scale-invariant* in the sense that basic objects with precisely the same shape but of different sizes have exactly the same prior probability. These probabilities are *shift-invariant* as well, so that the probability does not depend on the position of the object in the plane, either. We develop estimation algorithms — which we call *Scale-Invariant* algorithms — using MAP methods. These support vector estimators use prior knowledge about shape only; the position and size are estimated jointly, but using only that information which is provided in the measurements (in an ML fashion).

The first prior, called the Scale-Invariant Close-Min prior, favors objects whose boundaries are smooth in a mini-max sense. That is, those objects whose smallest discrete radius of curvature is large have higher prior probability. This prior is given by

$$p_{SICM}(h) = \frac{1}{z} \exp\left(\frac{1}{\tau} \min\{-q^{\mathrm{T}} c_1, \ldots, -q^{\mathrm{T}} c_M\}\right) , \tag{11}$$

where c_i, $i = 1 \ldots M$, are the columns of the matrix C. The Scale-Invariant Closest prior given below, favors objects which are more circular in their overall shape and is given by

$$p_{SIC}(h) = \frac{1}{z} \exp\left(-\frac{1}{\tau}(q-e)^{\mathrm{T}}(q-e) \right) . \tag{12}$$

Finally, the Scale-Invariant Max-Area prior, favors objects whose area to squared circumference ratio is large. This prior is given by

$$p_{SIMA}(h) = \frac{1}{z} \exp\left(\frac{1}{\tau} S(q) \right) , \tag{13}$$

where $S(q) = -q^{\mathrm{T}}Cq/\tan\theta_0$ is the area of the basic object corresponding to the shape vector q. In each case, τ is a parameter of the probability and z is a constant chosen so that the probability integrates to one over a large M-dimensional ball centered at the origin. Each of these priors uses the knowledge that the true objects tend to be circular, which may be suitable in a wide variety of applications, but the manner in which this knowledge is specified is different in each case.

In order to specify the actual estimation algorithm we assume that each observed support value y_i is given by $y_i = h_i + n_i$ where h_i is the true support value and n_i is a zero-mean Gaussian random variable with variance σ^2 which is independent of all h_i and all n_j, $j \neq i$. The Scale-Invariant algorithms solve the MAP problem for each of the priors given above. The computations which are required have at their core a quadratic program (QP) which computes $\hat{q}$, the optimal shape vector. The QP arises due to the quadratic form of the priors and the Gaussian measurement probability, and to the linear support vector constraint. A variation on a standard line search algorithm is also required to find the optimal size $\hat{t}$; the optimal position $\hat{h}_n$ can, in general, be determined directly (without iteration) from the measurements. With minor modification, the Scale-Invariant Algorithms can be adapted for the cases of limited- and sparse-angle observations.

5 Ellipse-Based Algorithms

This section develops the use of prior knowledge about the eccentricity and orientation of the true object. We develop three Ellipse-Based algorithms, each of which uses this type of prior knowledge in a different way. The first algorithm finds the closest ellipse to an observed support vector, so that under the Gaussian noise model given above this algorithm may be used to find the ML estimates of the ellipse parameters, assuming the true object to be exactly an ellipse. The second algorithm assumes that the true object is nearly elliptic in shape and that we have some prior knowledge of the true ellipse parameters. Finally, the third algorithm assumes that the true object is nearly elliptic in shape, but that we do *not* have any prior knowledge of the true ellipse parameters. This algorithm estimates *jointly* the support vector and the ellipse parameters.

The key to the Ellipse-Based Algorithms is the characterization of the support vector of an ellipse $h(v, t, \varepsilon, \phi)$ where v is the 2-D vector indicating the center position of the ellipse,

t is the size of the ellipse (proportional to its circumference), ε is the eccentricity, and ϕ is the orientation. A formula for the ith element of $h(v,t,\varepsilon,\phi)$ is given by [4]

$$h_i = \frac{Mt\sqrt{\frac{1}{1-\varepsilon^2}\cos^2(\theta_i-\phi)+\sin^2(\theta_i-\phi)}}{\sum_{j=1}^{M}\sqrt{\frac{1}{1-\varepsilon^2}\cos^2(\theta_j-\phi)+\sin^2(\theta_j-\phi)}} + [\cos\theta_i \quad \sin\theta_i]v \ . \tag{14}$$

The ML estimate of the ellipse parameters — given the Gaussian noise model described in the previous section — may be found using the following optimization problem:

$$\begin{aligned} &\underset{v,t,\varepsilon,\phi}{\text{minimize}} && \|y-h(v,t,\varepsilon,\phi)\|^2 \\ &\text{subject to} && t\geq 0 \quad \text{and} \quad 0\geq\varepsilon\geq 1\,, \end{aligned} \tag{15}$$

where $\|x\|^2$ denotes $x^{\mathrm{T}}x$. Because of the complex form of $h(v,t,\varepsilon,\phi)$, this problem is highly nonlinear and must be solved iteratively. Since the objective function is differentiable and the interesting solutions do not lie on the constraints, we use the conjugate gradient method to solve this problem [4].

If we knew *a priori* that the true object shape is nearly elliptic with eccentricity $\bar{\varepsilon}$ and orientation $\bar{\phi}$, then our reconstruction algorithm should favor shapes which resemble the ellipse with these parameters. This knowledge can be included by specifying the Ellipse-based Scale-Invariant Closest (ESIC) prior as

$$p_{ESIC}(q) = \frac{1}{z}\exp\left\{-\frac{1}{\tau}\left\|q-h(0,1,\bar{\varepsilon},\bar{\phi})\right\|^2\right\}. \tag{16}$$

This prior is identical in form to the SI Closest prior except that the largest probabilities are concentrated around the ellipse whose support *shape* vector is $h(0,1,\bar{\varepsilon},\bar{\phi})$ rather than the shape vector $e=[1\,1\,\ldots\,1]^{\mathrm{T}}$, which is the support vector of the unit circle. The resulting MAP estimation algorithm performs a line search in t, solving a QP at each stage, until the jointly optimum $(\hat{t},\hat{q})$ pair is determined.

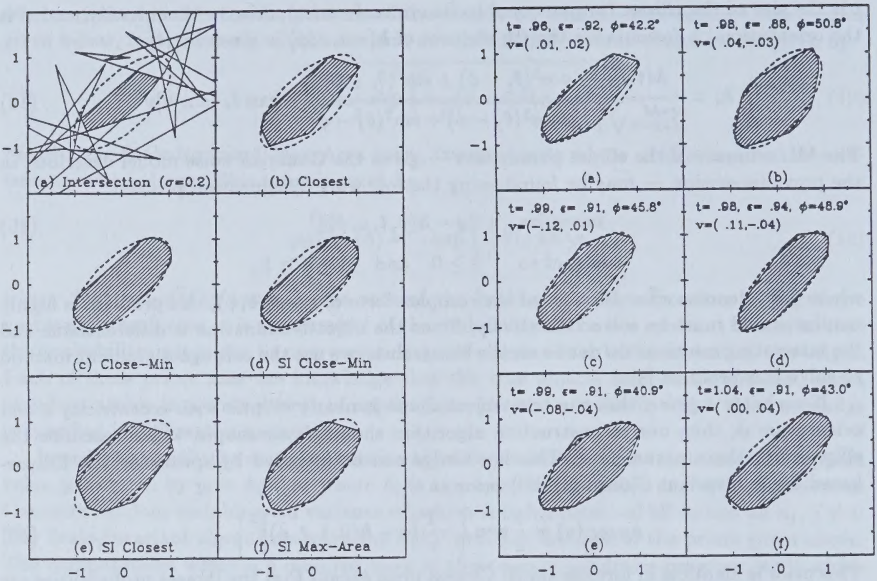
Next we extend these two algorithms to estimate *jointly* a support vector and ellipse parameters. Here, the prior knowledge we utilize is that the true support vector is likely to be near to the shape of an ellipse, *but we do not know which ellipse* a priori. This problem may be written formally as

$$\begin{aligned} &\underset{h,v,t,\varepsilon,\phi}{\text{minimize}} && \alpha\|h-y\|^2+(1-\alpha)\|h-h(v,t,\varepsilon,\phi)\|^2 \qquad 0<\alpha\leq 1 \\ &\text{subject to} && t\geq 0\,, \quad 0\leq\varepsilon\leq 1\,, \quad \text{and} \quad h^{\mathrm{T}}C\leq 0\,. \end{aligned} \tag{17}$$

We note that if $\alpha=1$ then the objective function of (17) is independent of the ellipse parameters[1] v, t, ε, and ϕ, and the optimum h is found using the Closest algorithm described in [5]. Alternatively, as $\alpha\rightarrow 0$, the optimum ellipse parameters approach their ML estimates, and the optimum h approaches the ellipse support vector corresponding to the ML estimates of the ellipse parameters. These two extremes provide some insight as to the solution in the general case when $0<\alpha<1$. For example, the optimum h cannot be closer to y than $\hat{h}_C$, the Closest estimate, since then it would be infeasible; also, it cannot be farther away from y than $h(\hat{v}_{ML},\hat{t}_{ML},\hat{\varepsilon}_{ML},\hat{\phi}_{ML})$.

---

[1] Here, v and t are *not* SSS coordinates of h.



(a) (b)

Figure 3: Comparison of several reconstruction algorithms.

6 Results

In Fig. 3, we present the results of two simulations which are designed to show the general behavior of the Scale-Invariant algorithms and the Ellipse-Based algorithms. In Fig. 3a, six different reconstruction methods are run on the same 30 support line observations ($\sigma = 0.2$) of the true ellipse. The true ellipse — characterized by parameters $t = 1.0$, $\varepsilon = 0.9$, $\phi = 45.0$, and $v = (0.0, 0.0)$ — is shown using dotted lines in each of the panels; the support line measurements are indicated by the solid lines in panel (a). The intersection method, which simply intersects the halfplanes defined by each of the support line measurements, produces the shaded result in panel (a). An ML algorithm (described in [5]) yields the result shown in panel (b). Panels (d), (e), and (f) show the result of the three Scale-Invariant algorithms described above for $\tau = 0.1$ and panel (c) is the result of the Close-Min algorithm, which is a simple variant of the Scale-Invariant Close-Min algorithm and is also described in [5].

It is clear that a great deal of improvement over the set intersection method and the ML method may be achieved using the Scale-Invariant (SI) algorithms, even in cases such as this where the noise variance is very large. The SI Close-Min estimate shown in panel (d) of Fig. 3a shows evidence that the prior knowledge is related to the boundary of the object. In particular, the estimate has a boundary without any of the sharp bends that are evident in the ML solution of panel (b). The SI Closest result shown in panel (e) has an overall shape that is more circular than any of the other estimates; this reflects the prior probability which favors objects which are close to the shape vector e. The SI Max-Area result shown in panel (f) has few long straight boundary sections, which reflects the tendency to increase the ratio of area to circumference squared; this property is the basis of the SI Max-Area prior.

Fig. 3b shows the results of 6 independent experiments which take a set of 30 observed support lines with the same noise variance as in Fig. 3a and apply the joint support vector/ellipse estimation algorithm with $\alpha = 0.5$ to jointly estimate a support vector and ellipse parameters. The estimated ellipse parameters are indicated in each panel and should be compared to the true values given above. The ellipse which corresponds to the estimated parameters is drawn in each of the panels using dashed lines. The estimated support vector yields a basic object which is shown as a shaded region in each of the panels.

For comparison, the support line observations used in panel (a) of Fig. 3b are the same as those used for all of the panels in Fig. 3a. In fact, the estimated ellipse-based support vector in panel (a) of Fig. 3b is similar in appearance to the ML estimate shown in panel (b) of Fig. 3a except that the sharp corners have been smoothed out. This in agreement with the notion that the true object is nearly elliptic in shape; the degree to which this property is reflected in the estimate is controlled by the parameter α.

Additional results which demonstrate the performance of the algorithms for changes in noise variance σ^2, prior parameter τ, and which show the behavior of the Ellipse-Based Scale-Invariant algorithm and the Ellipse-Based Closest Ellipse algorithm may be found in [4]. Also included in [4] are comparisons of the various algorithms for the cases of sparse- and limited-angle observations.

7 Discussion

The addition of the prior shape knowledge described herein improves the performance of the support estimation algorithms over those given in [5] in the cases where the true objects are either nearly circular, or have large portions of nearly circular boundaries, or are nearly elliptic in shape. The extra knowledge also allows us to solve sparse- and limited-angle problems in which there are unobserved support values over possibly large angular ranges. In these cases, the missing support values are automatically interpolated by the algorithms using the available prior shape information. In CT, these geometric estimation algorithms may be used to estimate the convex hull of the object directly from the projections, and this estimate may be used in the reconstruction process as described in [4]. In the cases where the observed angles are restricted and the observations are noisy, this procedure dramatically improves the quality of the reconstructed images over those obtained by convolution backprojection.

References

[1] G. T. Herman. *Image Reconstruction from Projections.* Academic Press, New York, 1980.

[2] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[3] J. M. Humel. *Resolving Bilinear Data Arrays.* Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering, 1986.

[4] J. L. Prince. *Geometric Model-Based Bayesian Estimation From Projections.* PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering, January 1988.

[5] J. L. Prince and A. S. Willsky. *Reconstructing Convex Sets From Support Line Measurements.* Technical Report LIDS-P-1704, M.I.T. Laboratory for Information and Decision Systems, September 1987. (Submitted for publication to IEEE Pattern Analysis and Machine Intelligence.)

[6] D. J. Rossi and A. S. Willsky. Reconstruction from projections based on detection and estimation of objects–parts I and II: performance analysis and robustness analysis. *IEEE Trans. ASSP*, ASSP-32(4):886–906, 1984.

[7] J. S. Schneiter. *Automated Tactile Sensing for Object Recognition and Localization.* PhD thesis, Massachusetts Institute of Technology, Dept. of Mechanical Engineering, 1986.

[8] P. L. Van Hove. *Silhoutte Slice Theorems.* PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering, 1986.

[9] P. L. Van Hove and J. G. Verly. A silhouette-slice theorem for opaque 3-d objects. In *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Sig. Proc.*, pages 933–936, IEEE ASSP, March 26–29 1985.

A TWO-DIMENSIONAL AUTOREGRESSIVE MODELING TECHNIQUE FOR ANALYSIS AND CLASSIFICATION OF PLANAR SHAPES

MANOHAR DAS, MARK J. PAULIK, N. K. LOH
Center for Robotics and Advanced Automation, Oakland University Rochester, MI 48309

ABSTRACT

A two-dimensional vector autoregressive model is introduced for the analysis and classification of closed planar shapes. The boundary coordinate sequence of a digitized binary image is sampled to produce a polygonal approximation to an object's shape. This circular sample sequence is then represented by a vector autoregressive difference equation which models the individual cartesian coordinate sequences as well as coordinate interdependencies. Several classification features which are functions or transformations of the estimated coefficient matrices and the associated residual error covariance matrices are developed. These features are shown to be invariant to object transformations such as translation, rotation and scaling. Laboratory experiments involving object sets representative of industrial shapes are presented. Superior classification results are demonstrated.

I INTRODUCTION

An important topic in computer vision and image processing is the classification of objects in a scene. It has been noted [1] that classification features based on the boundaries or edges of shapes are closely related to the salient features used by the human visual system for scene analysis. Examples of shape analysis based on boundary information can be found in military target identification [2-4], character recognition [5], x-ray diagnosis [6], industrial part identification [7-8], and scene analysis [9].

There are currently a large variety of methods available for the analysis of shape boundaries, but they may be loosely arranged into three groups: Space Domain Techniques [10-11], Scaler Transform Techniques [1],[3-7],[10],[12-17], and Curve fitting algorithms [8],[18-23].

In this paper, we explore a scalar transform method which is an extension of the methods based on one-dimensional (1-D) CAR models. A given shape is approximated by N samples of its original boundary sequence. The x and y cartesian coordinates of these samples referenced to the object centroid serve as the elements of a multidimensional spatial series. This vector series is then represented by a stochastic circular autoregressive model characterized by a set of unknown coefficient matrices and an independent vector noise sequence. Certain functions of these matrices are introduced and shown to be invariant to rotation, translation, and scaling.

A vector sequence representation has certain advantages over one-dimensional methods. Since both the x and y coordinates of the shape boundary are used, both convex and non-convex objects may be treated in the same fashion. Although the boundary unwrapping technique introduced by Dubois and Glanz [7] allows the extension of radii based methods to non-convex shapes, it loses boundary phase information. Such phase loss can allow different shapes to produce similar sequences. A 2-D boundary representation allows a variety of sampling schemes to be used. In all cases, the resulting vector sample sequence contains enough of the original boundary information to uniquely describe any simply connected planar shape. In the 1-D case, however, only the equal-angle radii sampling method [24], retains sufficient boundary information to uniquely describe a shape. Even then, only the limited class of shapes for which the radius vectors from the centroid intersect the boundary only once, can be represented uniquely. You and Jain [25] point out that all current shape classification schemes produce results which are somewhat data dependent. Because of 2-D sampling flexibility, it is sometimes possible, depending upon the data set, to choose a sampling method which biases the resulting spatial series so as to improve classification results. A presentation on the issues involved in the choice of sampling rate and method for shape analysis can be found in [24]. Finally, the vector method proposed in this paper allows the modeling of the x and y coordinate sequences as well as the inter-relationships between them. We have found that this results in models with lower residual errors than that achieved with a one-dimensional model (see Section III). Also, extension of this 2-D method to sequences described on higher dimensional lattices is straightforward. Other 2-D methods which provide context for our algorithm can be found in [3],[5],[15], and [21]. To our knowledge, the work presented in this paper constitutes the first attempt toward modeling and classification of planar shapes using vector CAR models.

The organization of the paper is as follows. Section II discusses the vector CAR model parameter estimation and the associated classification features. Section III presents experimental results and discussions while Section IV presents conclusions.

II BOUNDARY MODELING

In this section we describe the modeling of the boundary approximations. The CAR model coefficient matrices are estimated from a boundary sequence using a least squares estimation technique. These matrices and the model residual error are then used to derive classification features which are invariant to object translation, rotation and scaling.

A. Mathematical Modeling

Since the shape boundaries are closed contours, we have $X(i) = X(i+N)$, where $\{X(i), i=1,2,...N\}$ are coordinates of the boundary samples used to approximate the shape. We choose to fit the following circular autoregressive model:

$$Y(k) = \sum_{j=1}^{P} A_j Y(k-j) + \sqrt{\beta} V(k), \tag{1a}$$

$$X(k) = \alpha + Y(k) \tag{1b}$$

where $\{A_j, j = 1,2,..p\}$ are 2×2 coefficient matrices, and α is a 2×1 process mean vector, i.e., $E[X(k)] = \alpha$, where $E[\cdot]$ denotes the expected value of $[\cdot]$. The modeling error is represented by $\sqrt{\beta}V(k)$ and is taken as a zero mean white noise random vector sequence with β as the covariance matrix.

This model may also be written in state space form as follows:
Define the following matrices,

$$Y_s(k) = [Y^T(k),\ Y^T(k-1),\ Y^T(k-2),\ ...,\ Y^T(k-P+1)]^T, \tag{2a}$$

$$A_s = \begin{bmatrix} A_1 & A_2 \dots & & A_{p-1} & A_p \\ I_2 & 0 & \dots & 0 & 0 \\ 0 & I_2 & 0 \dots & 0 & 0 \\ 0 & 0 & I_2 \dots & 0 & 0 \\ 0 & 0 & \dots & I_2 & 0 \end{bmatrix}, \tag{2b}$$

$$X_s(k) = [X^T(k),\ X^T(k-1),\ X^T(k-2),\ ...,\ X^T(k-p+1)]^T, \tag{2c}$$

$$\alpha_s = [\alpha^T,\ \alpha^T,\ ...,\ \alpha^T]^T, \tag{2d}$$

$$V_s(k) = [V^T(k)\sqrt{\beta}^T,\ 0,\ 0,\ ...,\ 0]^T, \tag{2e}$$

then we have

$$Y_s(k) = A_s Y_s(k-1) + V_s(k), \tag{2f}$$

$$X_s(k) = \alpha_s + Y_s(k), \tag{2g}$$

where A_s is the 2p×2p system matrix, and I_2 is a 2×2 unit matrix.

In order for this model to be useful for shape classification, features which are invariant to changes in shape rotation, scaling, and translation must be identified. Notice that when an object rotates in the horizontal plane, the values of the sample sequence X(k) undergo an orthogonal transformation, i.e., X(k) becomes:

$$X'(k) = BX(k), \tag{3}$$

where B denotes a 2×2 orthogonal rotation matrix. For example, B could have the form:

$$B = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \tag{4}$$

where ϕ is the angle of rotation.

If in addition, the size of a shape changes, it can be represented as a multiplication of the coordinate sequence by a scaler c, i.e.,

$$\begin{aligned} X'_c(k) &= cX'(k), \\ &= cBX(k) \end{aligned} \tag{5}$$

where the subscript c indicates a scaled sequence.

Consider now, the relationship of the model for a scaled and rotated shape to that of the unaltered shape. The model in (1) could be written in terms of the rotated values as:

$$Y'_c(k) = \sum_{j=1}^{p} A'_{jc} Y'_c(k-j) + \sqrt{\beta'_c}V(k), \tag{6a}$$

$$X'_c(k) = \alpha'_c + Y'_c(k), \tag{6b}$$

where $\sqrt{\beta_c'}V(k)$ is a modified white noise sequence. Substitution of (5) into (6b) yields:

$$cBX(k) = \alpha'_c + Y'_c(k),$$

or

$$X(k) = \frac{1}{c} B_c^{-1} Y'(k) + \frac{1}{c} B_c^{-1} \alpha' \tag{7}$$

comparing (7) with (1b) we have:

$$Y(k) = \frac{1}{c} B^{-1} Y'_c(k), \tag{8a}$$

and

$$\alpha = \frac{1}{c} B^{-1} \alpha'_c. \tag{8b}$$

Substituting (8a) into (6a) we have:

$$cBY(k) = \sum_{j=1}^{P} cA'_{jc} BY(k-j) + \sqrt{\beta'_c}V(k), \tag{9}$$

or

$$Y(k) = \sum_{j=1}^{P} \left[B^{-1} A'_{jc} B\right] Y(k-j) + \frac{1}{c} B^{-1} \sqrt{\beta'_c}V(k). \tag{10}$$

Comparing (10) with (1a), we note the following relationships:

$$A_j = B^{-1} A'_{jc} B, \tag{11a}$$

and

$$\sqrt{\beta} = \frac{1}{c} B^{-1} \sqrt{\beta'_c}, \tag{11b}$$

that is, the coefficient matrices undergo an orthogonal similarity transformation, and the covariance matrix is rotated and scaled. Orthogonal similarity (unitary) transformations generate interesting invariant subspaces. Lemma 1 and 2 below present several such subspaces.

Lemma 1: Under an orthogonal similarity transformation, the sum of the squares of the elements of a matrix remain invariant. Proof: Given in the Appendix.

Lemma 2: Under similarity transformation, the trace, determinant, and eigenvalues of a square matrix remain invariant. Proof: See [26-27] for a proof.

We may now state the following Theorem:

Theorem 1: Consider a sequence $\{X(i), i=1,2,...N\}$, which is composed of samples from a simply connected closed planar shape boundary. If the sequence is modeled by the vector circular autoregressive model (CAR) given in (1) and (2), then the following features are invariant to translation, rotation, and scaling of the shape:

(a) the trace, determinant, eigenvalues, and elemental sum of squares of the coefficient matrices $\{A_j, j=1,2,...p\}$,
(b) the trace, determinant and eigenvalues of the system matrix A_s,
(c) the combination of the mean vector α and the covariance matrix β designated as tau: $\tau = \alpha^T\beta^{-1}\alpha$. Proof: Given in the Appendix.

B. Parameter Estimation

The CAR model parameters of (1) are estimated from the samples of the boundary sequence using a conventional least squares method [28]. The least squares estimates can be found as follows. Define,

$$\phi(k) = [Y^T(k-1),\ Y^T(k-2),\ \ldots,\ Y^T(k-P)]^T, \tag{12a}$$

and

$$\Theta = [A_1, A_2, A_3, \ldots A_P]^T. \tag{12b}$$

The predictor for the CAR model (1a) can be written as

$$\hat{Y}(k) = \Theta^T\phi(k), \tag{13}$$

and the quadratic norm for minimizing the prediction error is chosen as

$$J = \frac{1}{N}\sum_{k=1}^{N} \|Y(k) - \hat{Y}(k)\|^2. \tag{14}$$

The least squares estimate of the parameter vector Θ can be shown to be given by

$$\hat{\Theta}_{LS} = \left[\sum_{k=1}^{N} \phi(k)\phi(k)^T\right]^{-1} \sum_{k=1}^{N} \phi(k)Y^T(k). \tag{15}$$

The process mean vector and the model residual error covariance matrix can be found as

$$\hat{\alpha} = \frac{1}{N}\sum_{k=1}^{N} X(k), \tag{16}$$

and

$$\hat{\beta} = \frac{1}{N}\sum_{k=1}^{N} \left[Y(k) - \hat{\Theta}^T\phi(k)\right]\left[Y(k) - \hat{\Theta}^T\phi(k)\right]^T. \tag{17}$$

Theorem 1 identified several invariant features based on the CAR model coefficient matrices and the associated mean vector and covariance matrix. The actual parameters used must be estimated from the data sequence as indicated above. Theorem 2 below states that the estimation process yields parameters which maintain the same invariant characteristics.

Theorem 2: Let $\hat{\Theta}$, $\hat{\beta}$, and $\hat{\alpha}$ denote the least squares estimates of the CAR model parameters for a particular shape boundary sequence. The following features, based on these estimates, are invariant to translation, rotation, and scaling of the shape:

(a) the trace, determinant, eigenvalues, and elemental sum of squares of the estimated coefficient matrices $\{A_j, j=1,2,...p\}$,
(b) the trace, determinant and eigenvalues of the system matrix A_s,
(c) the following combination of the mean vector α and the error covariance matrix $\hat{\beta}$, $\hat{\tau} = \hat{\alpha}^T\hat{\beta}^{-1}\hat{\alpha}$. Proof: Given in the Appendix.

The intraclass invariant properties of the system matrix eigenvalues, the sum of squares of the elements of the coefficient matrices, and the mean vector - error covariance matrix product τ, make them excellent candidates as shape features for recognition purposes. Digitization errors, imaging noise, and parameter estimation errors are among the influences which can cause the feature vectors within the same class to exhibit some variations. These variations will typically generate close clusters in the feature space which can be adequately characterized by a variety of statistical pattern recognition techniques.

III EXPERIMENTS AND RESULTS

In this section we present experimental results which demonstrate the effectiveness of CAR modeling of shape sequences for object recognition. We treat nonoverlapping simply connected planar shapes which are completely known, i.e., no partial boundaries are considered. The images are binary, and the boundaries are located using a simple boundary follower technique [12]. Some random noise is expected due to the imaging system and quantization noise. Such images often arise in industrial situations involving robotic vision systems. The shapes used to demonstrate classification accuracy were taken from Dubois and Glanz [7]. This choice allows direct comparison of classification results and provide a means for illustrating the strengths of the technique developed in this paper. In addition, the 1-D CAR model of Kashyap and Chellappa [15] as well as Dubois and Glanz's extensions which unwrap the boundaries of non-convex shapes were implemented to allow our current results to be directly compared under the same data base.

A. Classification Feature Vector

As was indicated in Section II the vector CAR model yields coefficient matrices A_j which are not invariant to shape rotation. The task then is to identify functions or transformations which extract salient information from these matrices while remaining invariant to scaling and rotation. The state space formulation for the CAR model is attractive because the eigenvalues of the system matrix completely identify the system modes, and therefore are excellent feature candidates. Although the eigenvalues of the individual coefficient matrices could also be used as invariant features, they are not capable of completely describing the system modes. However, if a real feature vector is desired, the trace and determinant of the individual matrices can be used effectively as features.

The sum of squares of the elements of the coefficient matrices is used as a feature because it exploits the fact that object rotation results in a unitary transformation rather than just a similarity transformation. A conventional similarity transformation would result in invariant eigenvalues corresponding to an equivalent system, but the sum of squares feature would vary.

The feature τ, is important in that it is based on the process mean vector and on the model residual error which is an indication of model fit. An alternative way to incorporate the information in the residual error covariance matrix into the feature vector is to use the eigenvalues of the quantity $\beta/(\alpha^T\alpha)$ as features. This allows more detailed information about the quality of the CAR model fit to be used for classification at the cost of increasing the dimension of the feature vector by one. These eigenvalues can be shown to be invariant by following a procedure similar to that presented in the proof of Theorem 1. The actual feature vector used in this section can be written as $\langle \mathrm{eig}(\hat{A}_s), S(\hat{A}_k), \tau \rangle^T$ where $\mathrm{eig}(\hat{A}_s)$ denotes the eigenvalues of the estimated system matrix, and

$$S(\hat{A}_k) = \sum_{k=1}^{P} \sum_{i=1}^{2} \sum_{j=1}^{2} \left[\hat{a}'_{kij}\right] \left[\hat{a}'_{kij}\right], \qquad (20)$$

is the sum of the squares of the elements of the model coefficient matrices. Two statistical pattern classification algorithms were implemented to test the quality of the feature vectors derived from the 2-D CAR model: the Feature Weighting (FW) method and the Rotated Coordinate System (RCS) method [29].

B. Data Acquisition and Description

The shapes used for the experimental portion of this paper are illustrated in Figure 3, and were obtained from photographic enlargements of those presented by Dubois and Glanz [9]. Shape set A is composed of four convex shapes which are relatively similar and serve to test the sensitivity of the shape model to small variations in object shape. Shape set B contains eight different objects adapted from photographs of industrial shapes. These shapes are relatively complicated, and serve to test the performance of the 2-D CAR model when used to classify a larger number of classes.

The object photographs were digitized using a VICOM[1]/Sun-3 image processing system and 512 × 480 CCD camera. Thirty five images were taken for each object in set A. The objects were rotated by arbitrary angles, shifted within the image plane, and scaled by as much as 200 percent by changing the camera magnification. In a similar fashion, 50 images were taken for each object in set B. The resulting 256 gray level images were thresholded to produce binary images. The boundary sequences were obtained by a boundary follower algorithm.

In order to insure the accuracy of the image sample locations and eliminate perimeter dependence in the sampling algorithms, a polygonal approximation was calculated for each shape [19]. The resulting polygon was used to determine sample coordinates for both the 1-D and 2-D models. It was found that using samples calculated from the polygonal approximation rather than points taken directly from the boundary sequence resulted in lower overall variance in the feature vectors for both the 1-D and 2-D techniques. The equal-angle radii sampling scheme (Method 2) was used for the 1-D model while the equal curve length sampling technique (Method 1) was used for the 2-D model tests. For 2-D analysis, 64 boundary samples were used, and 64 equally spaced radii were used for the 1-D case.

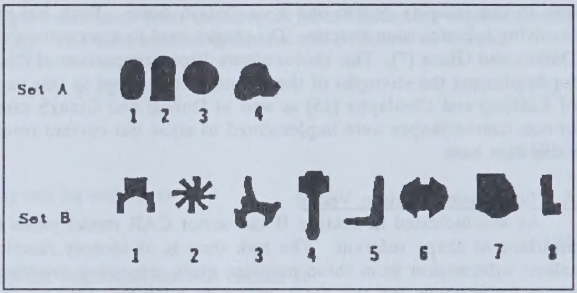


Fig. 1. Shape sets used for classification experiments.

C. Classification Results and Discussion

Table 1 presents the results of the classification tests for model orders one through four for the 2-D CAR model. Table 2 presents the 1-D results for comparison. Twenty five training samples were used for each object for both data sets A and B. Ten test samples were used for shape set A and 25 for shape set B. The model order was limited to 4 because complete 100% classification accuracy had been achieved by order 2. In addition, the use of higher model orders only serves to tax the quality of the intraclass feature vector covariance estimates, as well as the least squares parameter estimates.

TABLE 1

2-D CAR Model Shape Classification Results

Order	A RCS	A FW	B RCS	B FW
1	100%	92.5%	100%	100%
2	100%	100%	100%	100%
3	100%	100%	100%	100%
4	100%	100%	100%	100%

TABLE 2

1-D Shape Classification Results

Order	A RCS	A FW	B RCS	B FW
1	100%	85.0%	94.5%	78.5%
2	100%	82.5%	97.0%	86.5%
3	100%	82.5%	99.0%	82.0%
4	100%	90.0%	100%	85.0%

The results presented in Table 1 indicate that features based on the 2-D CAR model produced excellent classification results. Classification accuracy was 100% for all cases except the first order FW test which produced 3 errors out of the 40 samples tested. Comparison of Tables 1 and 2 illustrates that the 2-D model yielded features which resulted in superior classification. In particular, the FW method, which is relatively unsophisticated, yielded classification results very close to those of the RCS method. It is of interest to note that the feature vectors generated by the 2-D models are twice as large as their 1-D counterparts. Therefore, for a given model order, the 2-D method must estimate more parameters from the sampled boundary sequence. Also, the minimum number of training samples necessary to insure a non-singular covariance matrix estimate will be correspondingly larger. However, the results in Tables 1 and 2 show that the 2-D CAR modeling technique is capable of superior classification at significantly lower model orders. The determination of the best model order falls under the general topic of system identification and is treated in detail in the literature [28], [30-31].

An alternative way of quantifying the performance of the 2-D CAR model relative to the 1-D model is to examine the model prediction error. Equations (13), (14), and (19) indicate that the performance norm used to measure model quality is based on the ability of a model to forecast or predict the next sequence point given all past points. The model residual error sequence is formed as the difference between the actual boundary sample sequence and a sequence formed by one-step prediction values. The variance of such error sequences can be used as a measure of the quality of the model fit.

---

[1] VICOM is a registered trademark of VICOM Systems INC.

A signal to noise ratio (SNR) can be developed to provide a normalized measure of model performance. If we define prediction error sequences as the Euclidean distance between the one-step prediction points and the actual boundary points we can write:

$$e_{xy}(k) = \left[\left[x(k) - \hat{x}(k)\right]^2 + \left[y(k) - \hat{y}(k)\right]^2 \right]^{1/2}, \qquad (24)$$

and

$$e_r(k) = \left[\left[x_r(k) - \hat{x}_r(k)\right]^2 + \left[y_r(k) - \hat{y}_r(k)\right]^2 \right]^{1/2}, \qquad (25)$$

where $x_r(k)$ and $y_r(k)$ are the coordinates of the radii-boundary intersections and $e_{xy}(k)$ corresponds to the 2-D model, while $e_r(k)$ corresponds to the 1-D model. Then a SNR can be defined as

$$\mathrm{SNR} = 10\log_{10}\left[\frac{\left[\sigma_x^2 + \sigma_y^2\right]}{\sigma_e^2} \right]. \qquad (26)$$

Signal to noise ratios for three of the shapes shown in Figure 1 are presented in Table 3. In addition, Figure 2 presents a means of illustrating the modeling accuracy graphically for shape B2. Figure 2b, provides overlays of the sampled-shape boundary with a shape approx-imation consisting of the one-step prediction points for the 2-D model, i.e., the point sequences Y(k) and Y (k). Figure 2c presents similar overlays for the 1-D model. It is clear that the 2-D CAR model provides lower prediction errors and therefore, a better representation of the shape boundaries.

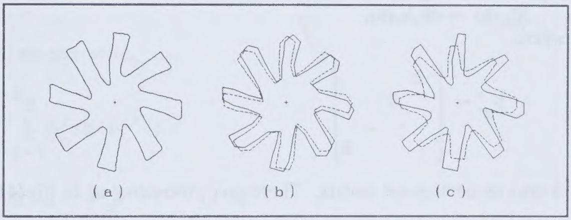


Fig. 2. Graphical illustration of model prediction accuracy: (a) Original shape B2. (b) Overlay of sampled boundary and one-step prediction sequence for 2-D model. (c) Overlay for 1-D model.

TABLE 3

Signal to Noise Ratios: First Order Model

Shape	SNR 2-D	SNR 1-D
B2	23.39	15.18
B3	26.62	16.12
B5	30.01	13.92

V CONCLUSION

In this paper, we introduced a new method which used a vector CAR model to analyze closed planar shapes. A 2-D circular autoregressive model of a sampled shape's boundary sequence was used to develop a set of classification features which were shown to be invariant to object translation, rotation, and scaling. Laboratory experiments comparing this new technique to the existing 1-D AR method demonstrated superior classification accuracy at significantly lower model orders.

APPENDIX

Proof of Lemma 1: Assume A to be an nxn matrix and let A be transformed by an orthogonal matrix U as follows:

$$A' = UAU^{-1}. \qquad (A\text{-}1)$$

Since $U^{-1} = U^T$ for orthogonal matrices, and tr(AB) = tr(BA) for any two matrices A and B with compatible dimensions, we obtain,

$$\sum_i \sum_j \left[a'_{ij}\right]^2 = \mathrm{tr}\left[A'\left[A'\right]^T\right]$$

$$= \mathrm{tr}\left[UAU^{-1}\left[UAU^{-1}\right]\right]$$

$$= \mathrm{tr}\left[UAA^{T}U^{T}\right]$$

$$= \mathrm{tr}\left[AA^{T}\right]$$

$$= \sum_{i}\sum_{j}\left[a'_{ij}\right]^{2}, \qquad \text{(A-2)}$$

where tr[·] denotes the trace of [·]. Q.E.D

Proof of Theorem 1: The shape sequences are invariant to translation because all coordinates are referenced to the object's centroid.

Consider a shape boundary sequence which has been subjected to both rotation by an orthogonal matrix B, and scaling by a scaler multiplier c, i.e.,

$X_c'(k) = cBX(k)$.

Noting the relationship between an unaltered sequence and one resulting from a rotated and scaled shape as given by (11a) we have by Lemmas 1 and 2 that the coefficient matrices A_j and A_{jc}' have the same trace, determinant, eigenvalues, and elemental sum of squares. This concludes proof of part (a).

To prove part (b) proceed as follows. Consider the state space model formulation of (2). We can represent scaling and rotation as follows:

$X'_{sc}(k) = cB_sX_s(k)$,

where

$$B_s = \begin{bmatrix} B & & & \\ & B & & \\ & & \cdot & \\ & & & B \end{bmatrix} \qquad \text{(A-3)}$$

is also an orthogonal matrix. Therefore, proceeding as in (6)-(11) we have,

$$X_s(k) = \frac{1}{c} B_s^{-1}\alpha'_{sc} + \frac{1}{c} B_s^{-1} Y'_{sc}(k) \qquad \text{(A-4)}$$

and therefore,

$$\alpha_s = \frac{1}{c} B_s^{-1}\alpha'_{sc}, \qquad \text{(A-5)}$$

$$Y_s(k) = \frac{1}{c} B_s^{-1} Y'_{sc}(k). \qquad \text{(A-6)}$$

Therefore, we have

$$Y_s(k) = B_s^{-1}A'_{sc}B_sY_s(k-1) + \frac{1}{c}B_s^{-1}V_s(k), \qquad \text{(A-7)}$$

$$A_s = B_s^{-1}A'_{sc}B_s. \qquad \text{(A-8)}$$

Applying Lemma 2 then, the trace, determinant, and eigenvalues of the system matrix are also invariant.

Finally, to prove part (c), consider τ for a scaled and rotated shape, defined as:

$$\tau'_c = \alpha'^{T}_{c}\beta^{-1}_{c}\alpha'_{c}. \qquad \text{(A-9)}$$

Since $\beta = \beta^T$ and $\rfloor\beta\rfloor\beta^T = \beta$, we obtain from (8b) and (11b),

$$\alpha'_c = cB\alpha, \qquad \text{(A-10)}$$

$$\sqrt{\beta}'_c = cB\sqrt{\beta}. \tag{A-11}$$

These yield:

$$[\beta'_c]^{-1} = [\sqrt{\beta}'_c\sqrt{\beta}'^T_c]^{-1} = \frac{1}{c^2}[B\sqrt{\beta}\sqrt{\beta}^T B^{-1}]^{-1}. \tag{A-12}$$

Therefore,

$$\begin{aligned}\alpha'^T_c\beta'^{-1}_c\alpha'_c &= c^2\alpha^T B^{-1}\frac{1}{c^2}[B\sqrt{\beta}\sqrt{\beta}^T B^{-1}]^{-1}B\alpha \\ &= \alpha^T B^{-1}B\beta^{-1}B^{-1}B\alpha \\ &= \alpha^T\beta^{-1}\alpha,\end{aligned} \tag{A-13}$$

that is,

$\tau'_c = \tau.$ Q.E.D.

Proof of Theorem 2: The estimates are invariant to translation because the data set coordinates are all referenced to the centroid of each object.

Consider a shape boundary sequence which has been subjected to both rotation by an orthogonal matrix B, and scaling by a scaler multiplier c, i.e.,

$X_c'(k) = cBX(k)$.

Then using the definition of $\mathbf{B}_s$ from (A-3) and $\phi(k)$ from (12a) we have,

$$\phi'_c(k) = cB_s\phi(k). \tag{A-14}$$

Substituting (8a) and (A-14) into (15) we can write,

$$\begin{aligned}\hat{\theta}'_c &= \left[\sum_{k=1}^{N}\phi'_c(k)\phi'_c(k)^T\right]^{-1}\sum_{k=1}^{N}\phi'_c(k)Y'^T_c(k), \\ &= B_s\left[\sum_{k=1}^{N}\phi(k)\phi^T(k)\right]B_s^{-1}B_s\left[\sum_{k=1}^{N}\phi(k)Y^T(k)\right]B^{-1} \\ &= B_s\hat{\theta}_{LS}B^{-1} \\ &= [B\hat{A}_1B_s^{-1}\ B\hat{A}_2B_s^{-1}\ldots,\ B\hat{A}_pB^{-1}]^T.\end{aligned} \tag{A-15}$$

Therefore, the estimated coefficient matrices undergo an orthogonal similarity transformation, and by Lemmas 1 and 2 we have that the trace, determinant, eigenvalues, and elemental sum of squares are invariant to rotation and scaling.

The trace, determinant, and eigenvalues of the estimated system matrix $\hat{\mathbf{A}}_s$ are shown to be invariant by replacing the actual component matrices of $\mathbf{A}_s$ with their estimates $\hat{A}_j$ to obtain $\hat{\mathbf{A}}_s$. Then, by (A-15) a rotated and scaled matrix estimate $\hat{\mathbf{A}}'_{sc}$ can be written as $\mathbf{B}_s\hat{\mathbf{A}}_s\mathbf{B}_s^{-1}$. Finally, by lemma 2 we have the desired invariant properties. This concludes proof of parts (a) and (b). Proof of part (c) follows.

The value of τ can be shown to be invariant in the same manner used in the proof of Theorem 1, equations (A-9)-(A-13). Using (16) and (17) we can write rotated and scaled versions of α and β as:

$$\begin{aligned}\hat{\beta}'_c &= \frac{1}{N}\sum_{k=1}^{N}\left[cBY(k) - cB\hat{\theta}^TB_s^{-1}B_s\phi(k)\right]\left[cBY(k) - cB\hat{\theta}^TB_s^{-1}B_s\phi(k)\right]^T, \\ &= \frac{c^2}{N}\sum_{k=1}^{N}B\left[Y(k) - \hat{\theta}^T\phi(k)\right]\left[Y(k) - \hat{\theta}^T\phi(k)\right]^T B_s^{-1}\end{aligned}$$

$$= c^{2}B\hat{\beta}B^{-1}, \qquad \text{(A-16)}$$

and

$$\hat{\alpha}'_{c} = \frac{c}{N}\sum_{k=1}^{N} BX(k),$$

$$= cB\hat{\alpha}. \qquad \text{(A-17)}$$

Substituting (A-16) and (A-17) into the expression for $\hat{\tau}$ we can write:

$$\hat{\alpha}'^{T}_{c}\hat{\beta}^{-1}_{c}\hat{\alpha}'_{c} = \hat{\alpha}^{T}B^{-1}\left[B\hat{\beta}B^{-1}\right]B\hat{\alpha},$$

$$= \hat{\alpha}^{T}\hat{\beta}\hat{\alpha}. \qquad \text{Q.E.D} \qquad \text{(A-18)}$$

REFERENCES

[1] C. T. Zhan and R. Z. Roskies, "Fourier descriptors for plane closed curves,", *IEEE Trans. Comput.*, vol. c-21, No. 3, pp. 269-281, Mar. 1972.

[2] S. A. Dudani et al., "Aircraft identification by moment invariants," *IEEE Trans. Comput.*, vol. c-26, pp. 39-46,Jan.1977.

[3] C. C. Lin and R. Chellappa, "Classification of partial 2-D shapes using fourier descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, No. 5, pp. 686-690, Sept. 1987.

[4] L. Gupta and M. D. Srinath, "Contour sequence moments for the classification of closed planar shapes," *Pattern Recognition,* vol. 20, No. 3, pp. 267-272, 1987.

[5] E. Persoon and K. S. Fu, "Shape discrimination using fourier descriptors," *IEEE Trans. Syst., Man, Cybern.*, vol., SMC-7, No. 3, pp. 170-179, Mar. 1977.

[6] D. H. Ballard and C. M. Brown, *Computer Vision*. New Jersey: Prentice-Hall, 1982.

[7] S. R. Dubois and F. H. Glanz, "An autoregressive model approach to two-dimensional shape classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, No. 1, pp. 55-66, Jan. 1986.

[8] W. A. Perkins, "A model-based vision system for industrial parts," *IEEE Trans. Comput.*, vol. c-27, No. 2, pp. 126-143, Feb. 1978.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[10] M. D. Levine, *Vision in Man and Machine.* New York: McGraw-Hill, 1985.

[11] T. Pavlidis, "Algorithms for shape analysis of contours and waveforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, No. 4, pp. 301-312, July 1980.

[12] R. C. Gonzalez and P. Wintz, *Digital Image Processing.* New Jersey: Prentice-Hall, 1985.

[13] A. Grattorola, S. Gaglio, L. Massone, and P. Morasso, "Coding shape details by means of frequency analysis of the boundary," *Proc. SPIE*, Vol. 594 on Image Coding, pp. 204-210, 1985.

[14] R. Chellappa and R. Bagdazian, "Fourier coding of image boundaries," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, No. 1, pp. 102-105, Jan. 1984.

[15] R. L. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Trans. Inform. Theory*, vol. IT-27, No. 5, pp. 627-637, Sept. 1981.

[16] P. F. Singer and R. Chellappa, "Classification of boundaries on the plane using stochastic models," *Proc. Conf. Comput. Vision Patt. Recog.*, pp. 146-147, June 1983.

[17] P. F. Singer and R. Chellappa, "Machine perception of partially specified planar shapes," *Proc. Conf. Comput. Vision Patt. Recog.*, pp. 497-502, June 1985.

[18] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol EC-10, pp.260-268, 1961.

[19] H. Freeman, "On the quantization of line-drawing data," *IEEE Trans. Syst., Sci., Cybern.*, vol. SSC-5, No. 1, pp. 70-79, Jan. 1969.

[20] T. Pavlidis, *Algorithms for Graphics and Image Processing.* Maryland: Computer Science Press, 1987.

[21] T. Galkowski and L. Rutowski, "New algorithm for shape analysis - The multidimensional case," *Proc. 8th ICPR*, 1984.

[22] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, "Two-dimensional, model-based, boundary matching using footprints," *Int. J. Robotics Res.*, vol. 5, No. 4, pp. 38-55, 1986.

[23] J. T. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves," *Int. J. Robotics Res.*, vol. 6, No. 2, pp. 29-44, 1987.

[24] M. Das, M. J. Paulik, and N. Loh, "A Two-Dimensional Autoregressive Modeling Technique for Analysis and Classification of Planar Shapes," *Oakland University Technical Report*, No. TR-CRAA-88-04, Center for Robotics and Advanced Automation, Oakland University, Rochester Michigan.

[25] Z. You and A. K. Jain, "Performance evaluation of shape matching via chord length distribution," *Comput. Graphics Image Processing*, vol. 28, pp. 185-198, 1984.

[26] F. R. Gantmacher, *The Theory of Matrices.* New York: Chelsea, 1977.

[27] G. Strang, *Linear Algebra and Its Applications.* New York: Academic Press, 1980.

[28] L. Ljung, *System Identification: Theory for the User.* New Jersey: Prentice-Hall, 1987.

[29] G. S. Sebestyen, *Decision-Making Processes in Pattern Recognition.* New York: Macmillan, 1962.

[30] G. E. P. Box and G. M. Jenkins, *Time Series Analysis Forecasting and Control.* San Francisco: Holden-Day, 1976.

[31] R. J. Bennett, *Spatial Time Series.* London: Pion Limited, 1979.

UNCERTAINTY AND SIMILARITY MEASURES IN PATTERN RECOGNITION

JOVAN DJ. GOLIĆ
Institute of Applied Mathematics and Electronics, Belgrade, Yugoslavia
Faculty of Electrical Engineering, University of Belgrade, Yugoslavia
School of Electrical Engineering, Cornell University, Ithaca, NY 14853

Abstract: Some general properties of feature evaluation criteria in statistical pattern recognition, including uncertainty, similarity, distance, and dependence measures, are pointed out. The relationship between the uncertainty and similarity measures is established. In addition, the tightest upper and lower bounds in terms of the Bayes probability of error are derived for the generalized divergence and for the generalized Matusita's affinity.

I. INTRODUCTION

Procedures for feature evaluation play a major role in a pattern recognition system design. They are generally aimed either at achieving as high a classification accuracy as possible, given a number of features, or at achieving a given classification accuracy with as small a number of features as possible. There are several reasons for such objectives. First, it is always desirable to discard irrelevant or redundant features, that is, those whose contribution to the classification accuracy is negligible. Second, the smaller the number of features the lower the complexity of both the classification and data acqusition stages of a pattern recognition system. Third, in theory, when the complete knowledge of the probability structure is assumed, the smaller the number of features the lower the classification accuracy of the optimal, Bayes classifier and, most often, of any reasonable classifier as well. However, in practice, the probability structure of a pattern recognition system is usually not known and, therefore, a classifier must be trained on a finite sample set, which, due to estimation variance and bias, may reduce its reliability and accuracy. Accordingly, a smaller number of features, which requires smaller training sets, often leads to more reliable a classifier and possibly to a more accurate one, as well.

There are basically two feature evaluation procedures: feature selection and feature extraction. The former one consists in selecting a feature subset, of a given size, that yields the maximum classification accuracy, whereas the latter means selecting a feature set transformation, out of a given class of transformations, that leads to the maximum classification accuracy. The former one is a discrete optimization procedure, whereas the latter is usually a continuous one. Both of them may be optimal or suboptimal. Instead of a classification accuracy of a given classifier, as an optimization criterion, that is, as a feature evaluation rule one often chooses the maximum classification accuracy, namely, the accuracy of the Bayes classifier, since being independent of a classification rule it actually measures the discrimination information that is inherent to features only, not to a classifier as well. Accordingly, speaking in terms of the Bayes risk, the Bayes probability of incorrect classification or, simpler, of error, P_B, may be regarded as a reference feature evaluation criterion. However, in practice, P_B often performs worse than alternative feature evaluation criteria, for the following reasons.

First, as a functional of a pattern recognition system probability structure, it is not sensitive enough. Moreover, the greater the prior probability of the most probable class the more insensitive it is. As a consequence, even when the objective is to minimize P_B itself, in suboptimal procedures for feature selection or extraction, alternative rules, more sensitive for differentiating between feature subsets or transformations, usually perform better than P_B when used as optimization criteria. Appart from that, due to its insénsitivity, P_B is a poor measure of the discrimination information between classes when subsets of classes are considered instead of the whole set.

Second, P_B is generally suitable neither for analytical computation nor for estimation, which is required in both optimal and suboptimal procedures for feature selection or extraction. In addition, it does not enable recursive computation, which is very desirable in the sequential procedures for feature selection, optimal or suboptimal.

When a substitute rule is being used instead of P_B, a natural question arises as to their relationship. It is usually expressed in terms of the corresponding upper and lower bounds, although a more general approach, involving the similarity criteria between feature evaluation rules, is possible, see [1],[2]. Numerous feature evaluation rules, along with their interrelations, have been introduced and investigated. They are often classified under the three general categories: uncertainty, distance, and dependence measures, see [3],[4]. Essentially being real functionals of their probability structure, feature evaluation rules actually serve to compare pattern recognition systems, that is, to induce equivalence and preference relations among them. Therefore, they could be more generally called the efficiency measures, see [2].

This paper is basically intended to investigate the relationship among the general families of feature evaluation rules, to point out or establish their general properties, some of which are not known, and to derive some new bounds on the Bayes probability of error.

II. SIMILARITY MEASURES

In this Section we essentially outline some main definitions and results from [5], pertaining to the measures of separation or dissimilarity between probability measures. Let $\mu_1, \ldots, \mu_Q$, $2 \le Q < \infty$, be a finite set of probability measures on a measurable space $(X, \tilde{X})$ with the probability densities $p_1(x), \ldots, p_Q(x)$, $x \in X$, with respect to a dominating σ-finite measure ν. (Note that for a finite set of probability measures a dominating σ-finite measure always exists, e.g., $\nu = \sum_{i=1}^{Q} \mu_i$.) An f-dissimilarity of $\mu_1, \ldots, \mu_Q$ is defined by

$$D_f(\mu_1, \ldots, \mu_Q) = \int_X f(p_1(x), \ldots, p_Q(x))d\nu, \tag{2.1}$$

where $f(\vec{s})$, $\vec{s} \in \Sigma_Q$, is a convex, continuous, and homogeneous real function and $\Sigma_Q = \{\vec{s} = (s_1, \ldots, s_Q) \mid 0 \le s_i < \infty, i = 1, \ldots, Q\}$. Note that $f(0, \ldots, 0) = 0$. The dissimilarity function f could also be allowed to be an extended real function taking values in $(-\infty, +\infty]$. Alternatively, to a concave, continuous, and homogeneous real function $f(\vec{s})$, $\vec{s} \in \Sigma_Q$, there corresponds the so-called f-similarity $S_f(\mu_1, \ldots, \mu_Q)$. The similarity function f could also be allowed to be an extended real function taking values in $[-\infty, +\infty)$.

The fact that f is homogeneous brings us to the first general property of f-dissimilarities, see [6].

Property 2.1: An f-dissimilarity does not depend on the choice of a dominating σ-finite measure ν.

Employing the convexity of f one can prove the two following properties of f-dissimilarities, see [5].

Property 2.2: The f-dissimilarity $D_f(\mu_1, \ldots, \mu_Q)$ takes on its minimum value $f(1, \ldots, 1)$ if the probability measures $\mu_1, \ldots, \mu_Q$ are the same. It takes on its maximum value $f(1, 0, \ldots, 0) + f(0, 1, \ldots, 0) + \ldots + f(0, 0, \ldots, 1)$ if the probability measures are orthogonal, that is, if $\mu_i(A)>0 \to \mu_j(A)=0$, $j \ne i$, for all $A \in \tilde{X}$. If f is a strictly convex function, then these conditions are also necessary.

Property 2.3: Let T be a measurable transformation from $(X, \tilde{X})$ into a measurable space $(Y, \tilde{Y})$ and let $\mu_1^T, \ldots, \mu_Q^T$ be the probability measures on $(Y, \tilde{Y})$ induced by T. Then

$$D_f(\mu_1^T, \ldots, \mu_Q^T) \le D_f(\mu_1, \ldots, \mu_Q),$$

with equality if T is a sufficient statistic, that is, if there exist nonnegative $\tilde{Y}$-measurable functions $g_i(y)$, $y \in Y$, $i = 1, \ldots, Q$, and a nonnegative $\tilde{X}$-measurable function $h(x)$, $x \in X$, such that

$$p_i(x) = g_i(T(x))h(x) \quad \text{a.s.}\nu, \quad i = 1, \ldots, Q.$$

If f is a strictly convex function, then this condition is also necessary.

These three properties are the basic ones to be satisfied by any measure of dissimilarity or separation between the probability measures. While the homogeneity of f is also a necessary condition for Property 2.1, although being an important sufficient condition, the convexity of f is not a necessary one for Properties 2.2 and 2.3. The true meaning of convexity will be revealed in the next Section, in the context of the so-called generalized f-dissimilarities. Apart from that, in relation to the Bayes probability of error, the symmetry of f proves to be a natural additional constraint to be imposed on f.

Note that a pattern recognition system involves not only the probability measures, but also the prior probabilities assigned to them. Accordingly, in order to apply the concept of f-dissimilarity to pattern recognition systems, one should also incorporate the prior probabilities as well. One way to do so is given in Section 3, and some others are discussed in Section 4.

We proceed now by giving some examples of f-dissimilarities or f-similarities, the first of which relates to the well-known $\tilde{f}$-divergence. Namely, for $Q = 2$, a function $f(\vec{s})$, $\vec{s} \in \Sigma_2$, is a dissimilarity one if and only if it could be put under the form

$$f(s_1, s_2) = s_2 \tilde{f}\left(\frac{s_1}{s_2}\right), \quad s_1 > 0, s_2 > 0 \tag{2.2}$$

with $f(0, s_2) = s_2 \lim_{t \to 0+} \tilde{f}(t)$, for $s_2 \neq 0$, $f(s_1, 0) = s_1 \lim_{t \to 0+} t\tilde{f}(1/t)$, for $s_1 \neq 0$, and $f(0, 0) = 0$, where $\tilde{f}(t)$, $t \in (0, \infty)$, is a convex continuous real or extended real function. This means that for $Q = 2$, a separation measure is an f-dissimilarity if and only if it is an $\tilde{f}$-divergence [7]. The sufficiency part of this statement is known [6], whereas the necessity one is not. It is therefore not surprising that almost all the known measures of separation between two probability measures, including the Chernoff, Bhattacharyya, Matusita, generalized Matusita, and Kolmogorov distances, as well as the Bayes probability of correct classification itself, see [8], essentially belong to the class of $\tilde{f}$-divergences. Note that the well-known Kullback-Liebler divergence, directed or symmetric, is itself an extended real function. Since the symmetry condition $f(s_1, s_2) = f(s_2, s_1)$ translates into the condition $\tilde{f}(t) = t\tilde{f}(1/t)$, $t \in (0, \infty)$, an $\tilde{f}$-divergence is symmetric if and only if $\tilde{f}$ can be put under the form

$$\tilde{f}(t) = \tilde{g}(t) + t\tilde{g}\left(\frac{1}{t}\right), \quad t \in (0, \infty), \tag{2.3}$$

where $\tilde{g}(t)$, $t \in (0, \infty)$, is a convex continuous real or extended real function.

Another example of f-similarities, which was actually noted in [6], is the Matusita's affinity [13], with the similarity function

$$f(s_1, \ldots, s_Q) = \left(\prod_{i=1}^{Q} s_i\right)^{\frac{1}{Q}}, \quad \vec{s} \in \Sigma_Q. \tag{2.4}$$

Lastly, the generalized Bayesian distance [16], [17], [8] is an f-dissimilarity measure with

$$f(s_1, \ldots, s_Q) = \left(\sum_{i=1}^{Q} s_i^R\right)^{\frac{1}{R}}, \quad \vec{s} \in \Sigma_Q, \quad R \geq 1.$$

III. UNCERTAINTY MEASURES

A concept of uncertainty measures, associated with pattern recognition systems, was generally introduced in [9], where some of their general properties were established too. Besides a slightly modified version of the definition from [9], we will also employ the definition from [10], involving the natural symmetry and continuity conditions. Some new general properties of uncertainty measures will be derived and the relationship between uncertainty and f-similarity measures will be established.

Let $\tilde{P}$ denote an arbitrary pattern recognition system with Q, $2 \leq Q < \infty$, classes-categories, $C = \{c_1, \ldots, c_Q\}$, associated with a measurable space $(X, \tilde{X})$, which is often called a sample or observation space. There are two basic ways to define $\tilde{P}$. One is by $(X, \tilde{X}, \mu_1, \ldots, \mu_Q, \Pi)$, where $\mu_1, \ldots, \mu_Q$ are the so-called conditional probability measures on $(X, \tilde{X})$ associated with the categories and $\Pi = (\pi_1, \ldots, \pi_Q)$ is the vector of the prior probabilities of the categories. The other is by $(X, \tilde{X}, \mu, \hat{\Pi}(x))$, where μ is the so-called mixture probability measure on $(X, \tilde{X})$ and $\hat{\Pi}(x) = (\hat{\pi}_1(x), \ldots, \hat{\pi}_Q(x))$, $x \in X$, is the vector function of the posterior probabilities of the categories. Note that $\mu = \sum_{i=1}^{Q} \pi_i \mu_i$, and $\hat{\pi}_i(x)$, $x \in X$, is the Radon-Nikodym derivative of $\pi_i \mu_i$ with respect to μ, $i = 1, \ldots, Q$. A pattern recognition system is said to have Q categories if all the prior probabilities are positive.

A generalized F-uncertainty of $\tilde{P}$ is defined by

$$U_F(\tilde{P}) = \int_X F(\hat{\Pi}(x))d\mu \tag{3.1}$$

where the uncertainty function $F(P), P \in A_Q$, is a bounded Borel real function and $A_Q = \{P = (p_1, \ldots, p_Q) \mid 0 \le p_i \le 1,\ i = 1, \ldots, Q,\ \sum_{i=1}^{Q} p_i = 1\}$. Say that a generalized F-uncertainty is symmetric, continuous, concave, or strictly concave if F is symmetric, continuous, concave, or strictly concave, respectively. A concave generalized F-uncertainty will be referred to just as an F-uncertainty. Note that a concave F is automatically a bounded Borel function which is continuous on $\mathrm{int}A_Q$. Also, a continuous F is a bounded Borel function too. What underlies this definition is the fact that the Bayes probability of error is itself a symmetric continuous uncertainty

$$P_B(\tilde{P}) = \int_X p_B(\hat{\Pi}(x))d\mu \tag{3.2}$$

with the Bayes probability of error function

$$p_B(P) = 1 - \max\{p_1, \ldots, p_Q\}, \quad P \in A_Q. \tag{3.3}$$

The following are the basic properties of uncertainty measures.

Property 3.1: For a symmetric F-uncertainty

$$F(1, 0, \ldots, 0) \le U_F(\tilde{P}) \le F(\Pi) \le F(\frac{1}{Q}, \ldots, \frac{1}{Q}), \tag{3.4}$$

where: the minimum is achieved if the conditional probability measures are orthogonal, that is, if $\hat{\Pi}(x)$ is equal to a permutation of the vector $(1, 0, \ldots, 0)$ a.s.μ ; the maximum $F(\Pi)$ is achieved if the category and sample probability spaces are statistically independent, that is, if $\hat{\Pi}(x) = \Pi$ a.s.μ; and the overall maximum is achieved if the category and sample spaces are statistically independent and the categories are equiprobable. For a strictly concave symmetric F-uncertainty all of these conditions are also necessary. For P_B the condition for the overall maximum is also necessary. For any symmetric F-uncertainty satisfying $F(1/2, 1/2, 0, \ldots, 0) > F(1, 0, \ldots, 0)$, including P_B accordingly, the condition for the minimum is also necessary.

Property 3.2: Let T be an arbitrary measurable transformation from $(X, \tilde{X})$ into a measurable space $(Y, \tilde{Y})$. Let $\tilde{P}$ be a pattern recognigion system associated with $(X, \tilde{X})$ and let $\tilde{P}^T$ denote the corresponding pattern recognition system induced by T. Then for an arbitrary F-uncertainty

$$U_F(\tilde{P}^T) \ge U_F(\tilde{P}) \tag{3.5}$$

with equality if T is a sufficient statistic, that is, if there exists a $\tilde{Y}$-measurable vector function $G(y), y \in Y$, such that

$$\hat{\Pi}(x) = G(T(x)) \quad \text{a.s.}\mu.$$

If F is a strictly concave function, then this condition is also necessary.

Property 3.3: For a given continuous generalized F-uncertainty, inequality (3.5) is satisfied for all measurable transformations T and all pattern recognition systems with Q categories if and only if the uncertainty function F is concave. For a given generalized F-uncertainty, inequality (3.5) is satisfied for all measurable transformations T and all pattern recognition systems with Q or fewer categories if and only if F is concave.

Property 3.4: Let T be an arbitrary transformation from the set of categories C into a finite set C'. Let $\tilde{P}$ be a pattern recognition system associated with C, and let ${}^T\tilde{P}$ denote the corresponding pattern recognition system, associated with C', induced by T. Then for an arbitrary symmetric F-uncertainty

$$U_F({}^T\tilde{P}) \le U_F(\tilde{P}) \tag{3.6}$$

with strict inequality if F is a strictly concave function and T is not a one-to-one mapping.

The meaning of all of these properties is self-evident. Property 3.1 can be proved by the straightforward application of the symmetry and concavity of F, along with the Jensen's inequality for unconditional expectations. The proof of Property 3.2 is based on the fact that the posterior probability vector function corresponding to $\tilde{P}^T$ is a conditional mathematical expectation of that corresponding to $\tilde{P}$, so that the Jensen's inequality for conditional expectations could be applied. Note that Property 3.3 essentially shows that the concavity is for Property 3.2 a necessary condition as well. It can be proved by observing the pattern recognition systems with only two points in the sample space. Finally, Property 3.4, showing that the uncertainty can not increase when the set of categories is partitioned, can be proved by employing the symmetry and concavity of F.

We now establish the connection between F-uncertainties and f-similarities. If in a pattern recognition system $\tilde{P}$, $p_1(x), \ldots, p_Q(x)$, $x \in X$, denote the probability densities of the conditional probability measures with respect to a dominating σ-finite measure ν, then the probability density of the mixture μ with respect to ν is given by

$$p(x) = \sum_{i=1}^{Q} \pi_i p_i(x), \quad x \in X \tag{3.7}$$

and the posterior probabilities by

$$\hat{\pi}_i(x) = \frac{\pi_i p_i(x)}{p(x)}, \quad x \in X, \, i = 1, \ldots, Q. \tag{3.8}$$

Accordingly, (3.1) can be put under the general form

$$U_F(\tilde{P}) = \int_X F\left[\frac{\pi_1 p_1(x)}{\sum_{j=1}^{Q} \pi_j p_j(x)}, \ldots, \frac{\pi_Q p_Q(x)}{\sum_{j=1}^{Q} \pi_j p_j(x)}\right] \sum_{j=1}^{Q} \pi_j p_j(x) d\nu, \tag{3.9}$$

that is,

$$U_F(\tilde{P}) = \int_X f(\pi_1 p_1(x), \ldots, \pi_Q p_Q(x)) d\nu \tag{3.10}$$

where

$$f(s_1, \ldots, s_Q) = F\left[\frac{s_1}{\sum_{j=1}^{Q} s_j}, \ldots, \frac{s_Q}{\sum_{j=1}^{Q} s_j}\right] \sum_{j=1}^{Q} s_j, \quad \vec{s} \in \Sigma_Q, \; \vec{s} \neq (0, \ldots, 0), \tag{3.11}$$

with $f(0, \ldots, 0) = 0$. Now, in order to relate $U_F(\tilde{P})$ to f-similarity measures, it remains to analyze under what conditions, regarding F, the function f, defined by (3.11), is concave, continuous, and homogeneous. The answer to this is given by the following simple mathematical result.

Proposition: A continuous homogeneous real function $f(\vec{s})$, $\vec{s} \in \Sigma_Q$, is concave if and only if its restriction to A_Q, that is, $F(P)$, $P \in A_Q$, is concave.

Namely, it is obvious that the concavity of f implies the concavity of its restriction F. On the other hand, it is not quite obvious that the concavity of F implies the concavity of its homogeneous extension f. What the Proposition essentially states is the following. Let for a pattern recognition system $\tilde{P}$ a generalized f-similarity be defined by

$$S_f(\mu_1, \ldots, \mu_Q, \Pi) = \int_X f(\pi_1 p_1(x), \ldots, \pi_Q p_Q(x)) d\nu, \tag{3.12}$$

which takes into account the prior probabilities as well, where the similarity function f is concave, continuous, and homogeneous. A generalized f-dissimilarity is defined analogously. This definition is in accordance with the examples given in [6]. Then, a continuous F-uncertainty and a generalized f-similarity are the same measure if $F(P)$, $P \in A_Q$, is the restriction of $f(\vec{s})$, $\vec{s} \in \Sigma_Q$.

With this equivalence in mind, it is clear that Property 3.1 partly follows from Property 2.2 and that Property 3.2 follows from Property 2.3 in the case of continuous uncertainty functions, although the proof in [5] is different from ours.

Some of the examples for symmetric and continuous F-uncertainties are the information [10], separability [11], and generalized Chernoff [12] measures, respectively defined by:

$$F(P) = \sum_{i=1}^{Q} f(p_i), \quad P \in A_Q,$$

where $f(z)$, $z \in [0, 1]$, is a concave and continuous real function;

$$F(P) = \sum_{i=1}^{Q} \sum_{j=i+1}^{Q} f(|p_i - p_j|), \quad P \in A_Q,$$

where $f(z)$, $z \in [0, 1]$, is a concave, continuous, and nonincreasing real function; and

$$F(P) = \sum_{i=1}^{Q} \sum_{\substack{j=1 \\ j \neq i}}^{Q} f(p_i^{\alpha} p_j^{1-\alpha}), \quad P \in A_Q, \ \alpha \in (0, 1/2],$$

where $f(z)$, $z \in [0, 1]$, is a concave, continuous, and nondecreasing real function.

Finally, applying the Proposition in the case $Q = 2$, we obtain that a function $f(\vec{s})$, $\vec{s} \in \Sigma_2$, is a similarity one if and only if it could be expressed by

$$f(s_1, s_2) = (s_1 + s_2)\tilde{f}\left(\frac{s_1}{s_1 + s_2}\right), \ (s_1, s_2) \neq (0, 0), \tag{3.13}$$

with $f(0, 0) = 0$, where $\tilde{f}(z)$, $z \in [0, 1]$, is a concave continuous real function. Moreover, f is a symmetric function if and only if $\tilde{f}(z) = \tilde{f}(1 - z)$, $z \in [0, 1]$, that is, if

$$\tilde{f}(z) = \tilde{g}(z) + \tilde{g}(1 - z), \quad z \in [0, 1] \tag{3.14}$$

where $\tilde{g}(z)$, $z \in [0, 1]$, is a concave continuous real function. This representation is simpler than that in terms of the $\tilde{f}$-divergence.

IV. OTHER MEASURES

In Section 3 we have shown that continuous F-uncertainties are essentially the same as the generalized f-similarities. However, the prior probabilities can be taken into account in other ways as well and, consequently, f-similarities can also give rise to other efficiency measures of pattern recognition systems. The best-known examples are the so-called distance and dependence measures , see [3],[4]. Given a convex, continuous, and homogeneous real function $f(s_1, s_2)$, $(s_1, s_2) \in \Sigma_2$, they could respectively be defined by

$$J_f(\tilde{P}) = \sum_{i=1}^{Q} \sum_{j=1}^{Q} \pi_i \pi_j \int_X f(p_i(x), p_j(x)) d\nu \tag{4.1}$$

$$K_f(\tilde{P}) = \sum_{i=1}^{Q} \pi_i \int_X f(p_i(x), p(x)) d\nu \tag{4.2}$$

$$= \sum_{i=1}^{Q} \int_X f(\tilde{\pi}_i(x), \pi_i) d\mu. \tag{4.3}$$

As a dependence measure one could also define the mutual information

$$I_F(\tilde{P}) = F(\Pi) - U_F(\tilde{P}), \tag{4.4}$$

corresponding to an uncertainty function F.

Note that for a fixed number of categories, all three of these measures should be applied directly, namely, the greater they are the more efficient a pattern recognition system is. Accordingly, they share the general Property 2.3 of f-similarities, that is, they can not increase on a transformed sample space. They also possess Property 2.2, as to the minimum and maximum values. However, as for their behavior on a transformed set of categories, starting from (4.3) and using the convexity and homogeneity of f one can prove that K_f can not increase, which is in accordance with the notion of dependence but is contrary to what one could expect from a measure that should be applied directly. On the other hand, neither J_f nor I_F has the monotonicity property, that is, they may decrease or increase on a transformed set of categories. Since their behavior on a transformed set of categories is not appropriate, they should be used as efficiency measures of pattern recognition systems only when the set of categories is fixed.

In order to make them more matched to the Bayes probability of error, the distance and dependence measures should be defined as the symmetric generalized dissimilarities:

$$\bar{J}_f(\tilde{P}) = \sum_{i=1}^{Q}\sum_{j=1}^{Q}\int_X f(\pi_i p_i(x), \pi_j p_j(x))d\nu$$

$$= \sum_{i=1}^{Q}\sum_{j=1}^{Q}\int_X f(\hat{\pi}_i(x), \hat{\pi}_j(x))d\mu, \tag{4.5}$$

$$\bar{K}_f(\tilde{P}) = \sum_{i=1}^{Q}\int_X f(\pi_i p_i(x), p(x))d\nu$$

$$= \sum_{i=1}^{Q}\int_X f(\hat{\pi}_i(x), 1)d\mu. \tag{4.6}$$

Note that $\bar{J}_f$ actually coincides with the generalized $\tilde{f}$-divergence, see [8], [15], with $\tilde{f}$ and f being connected by (2.2), that is,

$$\bar{J}_f(\tilde{P}) = \sum_{i=1}^{Q}\sum_{j=1}^{Q}\int_X \hat{\pi}_i(x)\tilde{f}(\frac{\hat{\pi}_j(x)}{\hat{\pi}_i(x)})d\mu, \tag{4.7}$$

while $-\bar{K}_f$ is essentially an information measure.

V. RELATIONSHIP WITH THE BAYES PROBABILITY OF ERROR

For any $2 \le Q < \infty$ and an arbitrary symmetric and continuous F-uncertainty measure, the tightest upper and lower bounds in terms of the Bayes probability of error P_B were given in [10], by a general theorem pertaining to the set of all the pattern recognition systems with Q categories. We apply that theorem to the symmetric generalized $\tilde{f}$-divergence, with $\tilde{f}$ satisfying (2.3),

$$\bar{J}_{\tilde{f}}(\tilde{P}) = \int_X \sum_{i=1}^{Q-1}\sum_{j=i+1}^{Q} \hat{\pi}_i(x)\tilde{f}(\frac{\hat{\pi}_j(x)}{\hat{\pi}_i(x)})d\mu, \tag{5.1}$$

which is due to (2.3) practically equivalent to (4.7), and to the generalized Matusita's affinity, see (2.4), which is an uncertainty measure

$$M(\tilde{P}) = \int_X \left[\prod_{i=1}^{Q}\hat{\pi}_i(x)\right]^{\frac{1}{Q}} d\mu. \tag{5.2}$$

Theorem 5.1: For any $2 \le Q < \infty$ and $P_B \in [0, 1-1/Q]$,

$$\frac{Q-2}{2}\tilde{f}(1)P_B + (Q-1)(1-P_B)\tilde{f}(\frac{P_B}{(Q-1)(1-P_B)}) \le \bar{J}_{\tilde{f}} \le (Q-1)\tilde{f}(0) - \frac{Q(2\tilde{f}(0)-\tilde{f}(1))}{2}P_B. \tag{5.3}$$

Both bounds are the tightest on the set of all the pattern recognition systems with Q categories. The lower one is a convex nonincreasing function, whereas the upper is a linear nonincreasing function.

Theorem 5.2: For any $2 \le Q < \infty$ and $P_B \in [0, 1-1/Q]$,

$$\max\{0, (Q-1)P_B-(Q-2)\} \le M \le (\frac{P_B}{Q-1})^{\frac{Q-1}{Q}}(1-P_B)^{\frac{1}{Q}}. \tag{5.4}$$

Both bounds are the tightest on the set of all the pattern recognition systems with Q categories. The lower one is a convex, piecewise linear nondecreasing function, whereas the upper is a concave increasing function.

Finally, we discuss the known results pertaining to these bounds. The upper bound in (5.3) was derived in [8] for $Q = 2$, while the lower one was established in [14] for $Q = 2$ and the Kullback-Leibler divergence ($\bar{f}(t) = (1-t)\log t$). The upper bound in (5.4) was derived in [15], although in a different way. All the other cases are covered by Theorems 5.1 and 5.2.

REFERENCES

[1] M. Ben-Bassat, "ε-equivalence of feature selection rules," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 769-772, Nov. 1978.

[2] Jovan Dj. Golić, "On the relationship between the efficiency measures of multicategory information systems," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 531-538, July 1987.

[3] M. Ben-Bassat, "Use of distance measures, information measures, and error bounds in feature evaluation," in *Handbook of Statistics*, *Vol.* 2. New York: North-Holland, 1982.

[4] P. A. Devijver and J. Kittler, *Pattern Recognition*: *A Statistical Approach*. Englewood Cliffs: Prentice Hall, 1982.

[5] L. Gyorfi and T. Nemetz, "f-dissimilarity: a generalization of the affinity of several distributions," *Annals of Institute of Statistical Mathematics*," vol. 30, pp. 105-113, 1978.

[6] L. Gyorfi and T. Nemetz, "f-dissimilarity: a general class of separation measures of several probability measures," in *Proc. 16th Coll. Inform. Theory*, Keszthely, Hungary, 1975.

[7] I. Csiszar, "Information-type measures of divergence of probability distributions and indirect observations," *Studia Sci. Math. Hungar.*, vol. 2, pp. 299-318, 1967.

[8] D. E. Boekee and J. C. A. van der Lubbe, "Some aspects of error bounds in feature selection," *Pattern Recognition*, vol. 11, pp. 353-360, 1979.

[9] M. DeGroot, "Uncertainty, information and sequential experiments," *Ann. Math. Statist.*, vol. 33, pp. 404-419, 1969.

[10] Jovan Dj. Golić, "On the relationship between the information measures and the Bayes probability of error," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 681-693, Sept. 1987.

[11] Jovan Dj. Golić, "On the relationship between the separability measures and the Bayes probability of error," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 694-701, Sept. 1987.

[12] Jovan Dj. Golić, "Chernoff measures, information measures, and probability of error," in *Proc. 22nd Conf. Inform. Scien. and Syst.*, Princeton, NJ, 1988.

[13] K. Matusita, "Some properties of affinity and applications," *Annals of Institute of Statistical Mathematics*, vol. 23, pp. 137-155, 1971.

[14] G. T. Toussaint, "On the divergence between two distributions and the probability of misclassification of several decision rules," in *Proc. 2nd Int. Joint Conf. Pattern Recognition*, Copenhagen, Danmark, 1974.

[15] G. T. Toussaint, "Probability of error, expected divergence, and the affinity of several distributions," *IEEE Trans. Systems*, *Man*, *Cybern.*, vol. SMC-8, pp. 482-485, June 1978.

[16] G. T. Toussaint, Feature evaluation criteria and contextual decoding algorithms in statistical pattern recognition, Ph.D. thesis, Dept. Elect. Engineering, University of British Columbia, 1972.

[17] P. M. Trouborst, E. Backer, D. E. Boekee, I. J. Boxma, "New families of probabilistic distance measures," in *Proc. 2nd Int. Joint Conf. Pattern Recognition*, Copenhagen, Danmark, 1974.

A Model and Synthesis of Texture on A 3-D Surface

Yoonsik Choe and R. L. Kashyap

School of Electrical Engineering, Purdue University
West Lafayette, Indiana 47907

Abstract

To recover 3-D structure from a natural scene image, neither the Shape-from-shading nor the Shape-from-texture analysis is enough, because both information of radiance and texture coexist within the surface of natural scene. A new 3-D texture model is developed by considering the scene image as the superposition of a random texture image and a smooth shaded image. This smoothed surface is obtained by taking average of pixel intensities from natural image, and texture image can be generated by substracting the smoothed image from the original one. Then, Shape-from-shading technique is applied to estimate the orientation of surface. For describing a random texture image, the long-correlation periodic model is chosen because of its good performance in texture synthesis and its ability to represent the coarseness and the pattern of the surface at the same time. Then, a projection technique is developed to deal with this particular random model, which has a non-isotropically distributed texture pattern. For estimating the parameters, a hybrid method which uses both the least square and the maximum likelihood estimates is applied directly to the given intensity function. By using these parameters, the synthesized image is obtained to reconstruct the original image.

Keywords: Shape-from-shading, Shape-from-texture, Long-correlation periodic model, 3-D texture synthesis.

1. Introduction

An important task in the image processing area is the recovery of 3-D scene information from the 2-D image. The recovery of missing dimension can be achieved by the use of multiple views taken at different times, or from different directions of view. However, even a single image contains powerful cues as to object definition and shape.

3-D analysis of the image can be catagorized into two main catagories; Shape-from-shading, Shape-from-texture. Pioneering work on the inference of shape-from-shading was done by Horn[1] and his co-workers. They have used the reflectance map, which shows scene radiance as a function of the surface gradient and the distribution of light sources, to extract 3-D surface information from image data. However, this technique is based on the assumption that the imaging geometry, i.e., illumination direction, illuminant strength, etc., are known a priori. Instead of requiring prior information, Pentland[2] proposed the model of surface using umbilical points, i.e., points at which the two principal curvatures of surface are equal and the local shape of the surface is approximately spherical. Later, this analysis was improved by Lee and Rosenfeld[3], using only first derivatives instead of second derivatives, to reduce estimation errors.

On the other hand, some researchers have developed shape-from-texture analysis from different point of view. Since texture gradients behave like intensity gradients, the shape of a surface can be inferred from the pattern of a texture on the surface, by applying statistical texture analysis. General idea to extract 3-D structure information from texture was discussed by Witkin[4], Kender[5], and Stevens[6]. However, these techniques are limited to some specific planar surfaces and smoothly curved surface. It is because of that even if an image could be reliably partitioned into homogeneous regions corresponding to surfaces of objects, the 2-D statistical features used for region description would still limit recognition to known objects observed from viewer's direction.[7]

For describing a natural scene image, both the above approaches have their own limitations. Shape-from-shading technique is applicable only under the assumption that surface is smooth enough to have clear radiance information, while shape-from-texture technique requires the surface to be relatively complex so that texture information can be extracted. Thus, neither technique is suitable to recover 3-D structure from natural scene, because both informations of radiance and texture coexist within the surface of natural scene. Therefore, a robust technique is needed to handle this. Recently, Pentland[8] suggested to use a fractal scaling parameter to measure the coarseness of the surface, and applied this to represent the natural scene surface. However, this fractal model is not enough to represent the real 3-D texture image, because even though two surfaces are estimated to have same fractal scales, these surfaces can have quite different texture patterns.

In statistical texture analysis, it is well known that the model-based approach has many advantages over other approaches. Every model may have a few specific purposes, and the model need only have just enough significant detail to satisfy these purposes. Several different random texture models, e.g., Facet model[9], AR model[10], ARMA model[11], Fractional models[12],[13], etc., have been suggested for 2-D texture image up to date. Among them, Long-correlation periodic model[14] is most suitable for our goal, because of its good performance in texture synthesis and ability to represent the coarseness and pattern of surface at the same time. However, in 3-D texture case, it is extremely difficult to model texture image statistically well, because the texture pattern in the picture image will not be evenly distributed. Therefore, the special treatment is needed to deal with this problem.

In this paper, a new 3-D texture model is developed to represent the 3-D surface. A 3-D surface image is considered as the superposition of a texture image and a smooth shaded image. Long-correlation periodic model and Shape-from-shading analysis are applied to the texture image and shaded image, respectively. The smooth shaded image is obtained by taking average of intensities of the neighbor pixels, and then the texture image can be generated by substracting the smoothed image from the original image. However, general 2-D texture models do not fit this texture image, because first, since this image should represent the tilted and slanted surface, the texture pattern must not be distributed isotropically over the image, and secondly, at the same time, total expected value of intensities must be zero in order to be superposed with the smoothed image. A more detailed discussion will be followed in section 2. A new 3-D version of Long-correlation periodic texture model is developed for an arbitrarily tilted and slanted texture image, by projecting the tilted and slanted model to the viewer's direction.

For estimating parameters of this texture model, a hybrid method of least square and maximum likelihood estimations is used. This estimation method is based on an iterative procedure, and all the parameters are estimated directly from the given surface image. Finally, the 3-D surface image is synthesized with these parameters to reconstruct the original image.

2. Model of 3-D Surface Image

The surface shape is usually defined in terms of the viewer's coordinate system with axes l_1, l_2, l_3, having l_3 axis in the viewing direction. Then, the observed intensity function $z(l_1,l_2)$ of local shape of 3-D surface can be considered as the sum of a deterministic function $x(l_1,l_2)$, and a statistical random function $y(l_1,l_2)$, whose expected value is zero:

$$z(l_1,l_2) = x(l_1,l_2) + y(l_1,l_2) \tag{2.1}$$

and

$$E[z(l_1,l_2)] = x(l_1,l_2) \tag{2.2}$$

Thus, $x(l_1,l_2)$ can be simply estimated by smoothing, i.e., taking average of intensity values in the proper size of window. Then, $y(l_1,l_2)$ can be estimated by substracting this function from the original image $z(l_1,l_2)$. Fig.1 shows this superposition in a simple 2-D case. Note that these intensity functions do not represent the actual shape of image, because the shading variations are caused by changes in surface orientation relative to the illumination direction **L**. Therefore, $x(l_1,l_2)$ and $y(l_1,l_2)$ need to be projected to the illumination direction to extract the shape information.

Let σ_L, τ_L be the slant and tilt of illumination direction **L** in the coordination system with l_1, l_2, l_3. The direction **L** induces a coordinate system with axes i_1, i_2, i_3, and this is derived from the following transformation.

$$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} \cos\sigma_L\cos\tau_L & \cos\sigma_L\sin\tau_L & -\sin\tau_L \\ -\sin\tau_L & \cos\tau_L & 0 \\ \sin\sigma_L\cos\tau_L & \sin\sigma_L\sin\tau_L & \cos\sigma_L \end{bmatrix} \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \tag{2.3}$$

Here, i_3 is the illumination direction **L**, and l_3 is the viewing direction.

Surface normal **N** is another important direction of 3-D geometry. If we define a new coordinate system with axes, m_1, m_2, m_3, having m_3-axis in surface normal direction, these axes can be derived from the coordinate system of viewing direction, i.e., l_1, l_2, l_3, by using the similar coordination transform to above with the different values of tilt τ and slant σ. Fig.2 depicts the relations between these three different directions and four different tilt and slant parameters, σ_L, τ_L, σ, and τ.

Let $x'(m_1,m_2)$ and $y'(m_1,m_2)$ be a deterministic and a random functions, respectively, defined on the surface normal image plane, i.e., $m_3=0$. Then, $z'(m_1,m_2)$ defined as the sum of these two functions, is nothing but the intensity function z , observed from surface normal. Then, functions x' and y' can be also considered as the functions x and y, represented by this coordinate system. However, note that y' can not be the same function of y so that they may satisfy the superposition property. Fig.3 depicts the relations between these functions.

2.1 Model of The Deterministic Function x

As discussed before, function x' corresponds to the function x. Thus, if we can estimate the illumination direction and the surface normal from the smoothed image function x, the deterministic part of the 3-D surface, i.e., x', can be constructed from the coordinate transformation. Since the function x presents a smooth shaded image, the parameters σ_L, τ_L, σ, and τ can be estimated by Shape-from-shading analysis. The partial derivatives of the intensity function x, with respect to l_1, l_2, i.e., $\frac{\partial x}{\partial l_1}$ and $\frac{\partial x}{\partial l_2}$ can be calculated over the given image region by using the 3 x 3 derivative operator window. Then, with these known values, the parameters σ_L, τ_L, σ, and τ can be estimated from the following equations given in [2],[3], under the assumption that the shape of local regions is a part of sphere.

$$\tau^{(i)}{}_L = \tau^{(i)} = \arctan\left[\left(\frac{\partial \overline{x}^{(i)}}{\partial l_2}\right)/\left(\frac{\partial \overline{x}^{(i)}}{\partial l_1}\right)\right], \tag{2.1.1}$$

$$\sigma^{(i)} = \arcsin\left[\left(\frac{\partial \overline{x}^{(i)}}{\partial l_1}\right)^2 + \left(\frac{\partial \overline{x}^{(i)}}{\partial l_2}\right)^2\right], \tag{2.1.2}$$

$$\overline{x}^{(i)}(l_1,l_2) = \lambda\rho(\pi\cos\sigma^{(i)}{}_L - \sigma^{(i)}{}_L\cos\sigma^{(i)}{}_L + \sin\sigma^{(i)}{}_L)/\ 2(1 + \cos\sigma^{(i)}{}_L), \tag{2.1.3}$$

where $\frac{\partial \overline{x}^{(i)}}{\partial l_i}$ and $\left(\frac{\partial \overline{x}^{(i)}}{\partial l_i}\right)^2$ are the average values over the i-th region, and $\overline{x}^{(i)}$ is an average value over all points with the same tilt as the illumination direction in the region. Here, $\lambda\rho$ is a constant related to the brightness. Notice that l_1, l_2 are not the pixel positions in the image; they are related to the region patch.

This Shape-from-shading technique will not be considered in this paper.

2.2 Model of The Random Function y

Random function y', defined on the surface normal plane image (m_1,m_2), can be approximated by a random field model, which is isotropically distributed over the surface normal plane. Note that since this model is based on 2-D texture model, this will fit the plane surface more than other shape of surface. Thus, we apply this model to the local shape of image, because the local shape will be closer and closer to the plane surface when we take the smaller patch. Among the various random field models, Long-correlation periodic model is chosen for our goal, because this model represents the fractal scale and the pattern of texture image at the same time. Typical Long-correlation periodic model is as follows[14].

$$y'(m_1,m_2)=(1-2\cos\omega_1 z_1^{-1}+z_1^{-2})^{-\frac{c}{2}}(1-2\cos\omega_2 z_2^{-1}+z_2^{-2})^{-\frac{d}{2}}\varsigma(m_1,m_2) \qquad (2.2.1)$$

for $m_1,m_2 = 0,1,\ldots,N\text{-}1$.

The corresponding DFT of this function is

$$Y'(k_1,k_2) = \left(1-2\cos\omega_1 e^{-j2\pi\frac{k_1}{N}} + e^{-j4\pi\frac{k_1}{N}}\right)^{-\frac{c}{2}}\left(1-2\cos\omega_2 e^{-j2\pi\frac{k_2}{N}} + e^{-j4\pi\frac{k_2}{N}}\right)^{-\frac{d}{2}} W(k_1,k_2), \qquad (2.2.2)$$

where z_i is the delay operator associated with m_i, $\varsigma(m_1,m_2)$ is an i.i.d. Gaussian sequence, and $W(k_1,k_2)$ is the corresponding DFT.

Thus, if we can properly estimate the fractal scaling parameters, c & d, and the pattern parameters, ω_1 & ω_2 directly from the given intensity function $y(l_1,l_2)$, the random function defined on the viewing direction image plane, i.e., $l_3=0$, can be synthesized with these parameters. Note that the statistical model of function y can not be simply obtained by the coordinate transformation, because the expected values of both y and y' must be zeros over the planes to satisfy the superposition properties. Therefore, such a function $y(l_1,l_2)$ that satisfies this requirement can be obtained by projecting function $y'(m_1,m_2)$ to the viewer's image plane. Fig.3 shows this projection.

Then, since l_1 and l_2 are the rotated axes from m_1 and m_2 by amount of τ and σ, from eqn.(2.2.1) and the coordinate transformation

$$l_1 = m_1\cos\sigma\cos\tau + m_2\cos\sigma\sin\tau \qquad (2.2.3a)$$

$$l_2 = -m_1\sin\tau + m_2\cos\tau \quad , \qquad (2.2.3b)$$

we have

$$y(l_1,l_2) = (1-2\cos\omega_1 z'^{-1}_1+z'^{-2}_1)^{-\frac{c}{2}}(1-2\cos\omega_2 z'^{-1}_2+z'^{-2}_2)^{-\frac{d}{2}}\varsigma(l_1,l_2) \qquad (2.2.4)$$

where z'_1, z'_2 are the delay operators, corresponding to l_1 and l_2, respectively.

By the definition of DFT, $W'(k_1,k_2)$ corresponding to white noise sequence $\varsigma(l_1,l_2)$ is

$$W'(k_1,k_2) = \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1}\varsigma(l_1,l_2)\exp\left\{-j\frac{2\pi}{N}(m_1k_1+m_2k_2)\right\}$$

and, from (2.2.3a),(2.2.3b),

$$= \sum_{l_1=0}^{N-1} \sum_{l_2=0}^{N-1} \varsigma(l_1,l_2) \exp\left\{ -j\frac{2\pi}{N}\left[\left(\frac{\cos\tau}{\cos\sigma}k_1+\frac{\sin\tau}{\cos\sigma}k_2\right)l_1+(-\sin\tau k_1+\cos\tau k_2)l_2\right] \right\} \tag{2.2.5}$$

Thus, as in [15],

$$z'_1 = z_1^{\frac{\cos\tau}{\cos\sigma}} z_2^{\frac{\sin\tau}{\cos\sigma}} \tag{2.2.6a}$$

$$z'_2 = z_1^{-\sin\tau} z_2^{\cos\tau} \tag{2.2.6b}$$

Define

$$k'_1 = \frac{\cos\tau}{\cos\sigma}k_1 + \frac{\sin\tau}{\cos\sigma}k_2, \tag{2.2.7a}$$

$$k'_2 = -k_1\sin\tau + k_2\cos\tau \tag{2.2.7b}$$

Then,

$$W'(k_1,k_2) = W\left(\frac{\cos\tau}{\cos\sigma}k_1 + \frac{\sin\tau}{\cos\sigma}k_2,\ -k_1\sin\tau + k_2\cos\tau\right) \tag{2.2.8}$$

$$= W(k'_1,\ k'_2) \tag{2.2.9}$$

Therefore, with these relations, we can have the following model of $Y(k_1,k_2)$.

$$Y(k_1,k_2) \triangleq \text{DFT of } y(l_1,l_2)$$

$$= \left(1-2\cos\omega_1 e^{-j2\pi\frac{k'_1}{N}} + e^{-j4\pi\frac{k'_1}{N}}\right)^{-\frac{c}{2}} \left(1-2\cos\omega_2 e^{-j2\pi\frac{k'_2}{N}} + e^{-j4\pi\frac{k'_2}{N}}\right)^{-\frac{d}{2}} W(k'_1,k'_2) \tag{2.2.10}$$

Note that the fractal scaling parameters, c and d, remain same to the ones before projecting, because of the scale invariance property of theirs[18],[19].

3. Estimation of The Parameters c, d, ω_1, ω_2

The estimation of parameters in this projected Long-correlation periodic model can be done in frequency domain, modifying the techniques suggested in [16]. The modulus of DFT of noise sequence, $\{|W(k_1,k_2)|,\ k_1=0,1,\ldots,N-1,\ k_2=0,1,\ldots,N/2\}$, is a white sequence with the following Rayleigh density[20].

$$f_{|W(k_1,k_2)|}(W) = \begin{cases} \dfrac{2W}{\rho N^2} \exp\left\{\dfrac{-W^2}{\rho N^2}\right\} & ,\ W \geq 0 \\ 0 & ,\ \text{otherwise} \end{cases} \tag{3.1}$$

Also, $W'(k_1,k_2)$ has same Rayleigh density as $W(k_1,k_2)$, because the function W is a i.i.d. white sequence and, as discussed before, W' is a one-to-one correspondent function to W. Thus, the density function of W' will be

$$f_{|W'(k_1,k_2)|}(W') = \begin{cases} \dfrac{2W'}{\rho N^2} \exp\left\{\dfrac{-W'^2}{\rho N^2}\right\} & ,\ W' \geq 0 \\ 0 & ,\ \text{otherwise} \end{cases} \tag{3.2}$$

Now, set

$$s_{k'_1,k'_2} = \left|2\cos\left(\frac{2\pi k'_1}{N}\right) - 2\cos\omega_1\right|^{\frac{c}{2}} \left|2\cos\left(\frac{2\pi k'_2}{N}\right) - 2\cos\omega_2\right|^{\frac{d}{2}} \tag{3.3}$$

Then, from the definition of k'_i, (2.2.8), we can have

$$| Y(k_1,k_2) | = s_{k'_1,k'_2}{}^{-1} \; | W'(k_1,k_2) | \tag{3.4}$$

$$= s_{k'_1,k'_2}{}^{-1} \; | W(k'_1,k'_2) | \tag{3.5}$$

Thus, the density function of $| Y(k_1,k_2) |$ can be represented by following equations.

$$f_{| Y(k_1,k_2) |}(Y) = s_{k'_1,k'_2} \, f_{| W(k'_1,k'_2) |}(s_{k'_1,k'_2} Y(k_1,k_2)) \tag{3.6}$$

$$= \begin{cases} \dfrac{2s^2_{k'_1,k'_2} Y(k_1,k_2)}{\rho N^2} \exp \left\{ \dfrac{-s^2_{k'_1,k'_2} Y^2(k_1,k_2)}{\rho N^2} \right\} & , Y(k_1,k_2) \geq 0 \\ 0 & , \text{otherwise} \end{cases}$$

The estimation of parameters can be done by a hybrid method of Least Square and Maximum Likelihood estimations. For LS estimation, if parameter ω_1 and ω_2 are set, $\theta = (c,d,\alpha)^T$ can be estimated by minimizing the following cost function.

$$J(\theta,\omega_1,\omega_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} (\log | Y(k_1,k_2) | + \frac{c}{2} \log | 2\cos\frac{2\pi k'_1}{N} - 2\cos\omega_1 |$$

$$+ \frac{d}{2} \log | 2\cos\frac{2\pi k'_2}{N} - 2\cos\omega_2 | + \alpha)^2 \tag{3.7}$$

$$= \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} (\log | Y(k_1,k_2) | - \theta^T Z(k'_1,k'_2))^2 \tag{3.8}$$

where $\alpha = -E[\log | W'(k_1,k_2) |]$, $\theta = (c,d,\alpha)^T$, and

$$Z(k'_1,k'_2) = \begin{bmatrix} -\dfrac{1}{2}\log | 2\cos\dfrac{2\pi k'_1}{N} - 2\cos\omega_1 | \\ -\dfrac{1}{2}\log | 2\cos\dfrac{2\pi k'_2}{N} - 2\cos\omega_2 | \\ -1 \end{bmatrix} \tag{3.9}$$

Thus,

$$[\hat{c},\hat{d},\hat{\alpha}]^T = (\sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} Z(k'_1,k'_2) Z^T(k'_1,k'_2))^{-1} (\sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} Z(k'_1,k'_2) \log | Y(k_1,k_2) |) \tag{3.10}$$

Also, ML estimators of ω_1, ω_2 can be calculated by maximizing the log-likelihood function $L(Y;\theta,\omega_1,\omega_2)$ with $\hat{\alpha}$, estimated from above.

$$L(Y;\theta,\omega_1,\omega_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} \log | Y(k_1,k_2) | - \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} \log(\frac{\rho N^2}{2}) \tag{3.11}$$

$$+ \frac{c}{2} N \sum_{k_1=0}^{N-1} \log | 2(\cos\frac{2\pi k'_1}{N} - \cos\omega_1) | + dN \sum_{k_2=0}^{N/2} \log | 2(\cos\frac{2\pi k'_2}{N} - \cos\omega_2) |$$

$$- \frac{1}{\rho N^2} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N/2} | 2(\cos\frac{2\pi k'_1}{N} - \cos\omega_1) |^{c} \, | 2(\cos\frac{2\pi k'_2}{N} - \cos\omega_2) |^{d} \, | Y(k_1,k_2) |^2$$

where ρ is a variance of $\varsigma(l_1,l_2)$ and can be estimated by the following equation in mean square sense.

$$\hat{\rho} = \frac{1}{N^2} \exp \{ \gamma - 2\hat{\alpha} - \frac{\pi^2}{6N^2} \} \tag{3.12}$$

where γ is Euler's constant ($= 0.5772157$) [20].

Therefore, parameters c, d, ω_1, ω_2 can be estimated from above iterative method, initially setting the values of ω_1, ω_2 arbitrarily between $(0, \frac{\pi}{2})$.

The estimation procedures given in this paper can be summarized as follows:

1) From the natural scene image, obtain the smooth surface image by taking average of intensity values of neighboring pixels.
2) From Shape-from-shading analysis, estimate the surface orientation parameters, i.e., τ, τ_L, σ and σ_L to the partitioned local shape of this smooth surface image.
3) Generate the random field image by substracting the above smooth surface image from the original image.
4) Fit the modified Long-correlation periodic model to this random image function with parameters estimated in step (2).

4. Examples

A series of examples with both synthetic images and real images are considered. All the examples are focused on the local shape of a 3-D surface, because a small local shape is a unit of the total image, and this small area can be modelled by a random field model. Since our 3-D texture model is developed based on the 2-D texture model, it is obvious that our model is better suited to a plane surface. Also, note that the local shape will become much closer to the plane surface when we take the smaller patch.

This section will clude two different types of examples: The first is the parameter estimation of our 3-D texture model from a real tilted and slanted texture image, and the other one is the synthesis of texture pattern on the known 3-D surface structures.

4.1 Real Image

A real image of a part of a tree in a grass meadow is considered (Fig. 4(a)). This image is smoothed by a 9×9 smoothing window (Fig.4(b)), and a random field image is obtained by subtracting this smoothed image from the original surface image (Fig.4(c)). Parameters c, d, ω_1 and ω_2 of our 3-D texture model can be estimated directly from the viewing direction, with parameters τ and σ calculated from local shape analysis. However, this technique is not considered in this paper.

4.2 Synthesized Image

Two different texture patterns (Fig.5(a)) are synthesized on a plane surface, each tilted and slanted by $\pi/2$ (Fig.5(b)). Our projected long-correlation model is fit to this synthesized image. True values of ω_1, ω_2, c, d are 0.3, 0.3, 0.9, 0.9, respectively, in the first pattern and 0.2, 0.2, 0.3, 0.9, respectively, in the second. After applying our hybrid estimation algorithm, these values are estimated to be 0.3201, 0.3209, 0.8973, 0.8967 in the first pattern and 0.2134, 0.2173, 0.3232, 0.8846 in the second. Fig.5(c) shows the texture images reconstructed using these values. Therefore, these examples demonstrate our model's capability to deal with 3-D surfaces.

5. Conclusion

Our 3-D texture model has the following potential advantage, compared to other models. This model is suitable to represent the complicated shape of natural surface, since this model contains both the pattern parameters ω_1, ω_2 and the fractal scaling parameters c, d, that is, from the different values of c, d and ω_1, ω_2, this model can represent the direction of pattern or coarseness of surface.

References

[1] B. K. Horn, *Robot Vision*, the MIT Press, McGraw-Hill, 1986

[2] A. P. Pentland, "Local Shading Analysis," *IEEE Trans. Pattern Anal. Machine Intelligence*, Vol PAMI-6, 1984, pp170-187

[3] C-H Lee and A. Rosenfeld, "Improved Methods of Estimating Shape from Shading Using Light Source Coordinate System," *Artificial Intelligence 26*, 1985, pp125-143

[4] A. P. Witkin, "Recovering Surface Shape and Orientation From Texture," *Artificial Intelligence 17*, 1981, pp17-45

[5] J. R. Kender, "Shape From Texture: An Aggregation Transform that Maps A Class of Textures Into Surface Orientation," *Proc. 6th IJCAI*, 1979, pp475-480

[6] K. A. Stevens, "The Visual Interpretation of Surface Contours," *Artificial Intelligence 17*, 1981, pp47-73

[7] J. M. Tenenbaum, M. A. Fischler, and H. G. Barrow, "Scene Modeling: A Structural Basis for Image Description," *Computer Graphics and Image Processing*, Vol 12, 1980, pp407-425

[8] A. P. Pentland, "Shading in Texture," *Artificial Intelligence 29*, 1986, pp147-170

[9] T-C Pong, R. M. Haralick, and L. G. Shapiro, "The Facet Approach to Shape From Shading," *Proc. of Workshop on Computer Vision*, 1984, pp143-149

[10] R. Chellappa, and R. L. Kashyap, "Digital Image Restoration Using Spatial Interaction Models," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol ASSP-30, 1982, pp461-472

[11] R. L. Kashyap, "Characterization and Estimation in 2-D ARMA Models for Images," *IEEE Trans. Information Theory*, 1984

[12] A. P. Pentland, "Fractal-based Description of Natural Scenes," *IEEE Trans. Pattern Anal. Machine Intelligence*, Vol PAMI-6, 1984, pp661-674

[13] R. L. Kashyap and P. M. Lapsa, "Synthesis and Estimation of Random Fields Using Long-Correlation Models," *IEEE Trans. Pattern Anal. Machine Intelligence*, Vol PAMI-6, 1984, pp800-809

[14] K-B. Eom, "Robust Image Models With Application," Ph.D Thesis, Purdue University, West Lafayette, Indiana, 1986

[15] H. Kang and J. K. Aggarwal, "Design of Two-Dimensional Recursive Filters by Interpolation," *IEEE Trans. Circuit and Systems*, Vol CAS-24, 1977, pp281-291

[16] R. L. Kashyap and K-B. Eom, "Estimation in Long-Memory Time Series Model," *J. of Time Series Analysis*, Vol. 9, 1988, pp35-41

[17] R. L. Kashyap, and K-B Eom, "Rubust Image Models and Their Applications," *Advances in Electronics and Electron Physics*, Vol 70, pp79-157

[18] J. R. M. Hosking, "Fractional Differencing," *Biometrika*, Vol 68-1, 1981, pp165-176

[19] R. L. Kashyap, "Image Models," *Handbook of Pattern Recognition And Image Processing*, pp281-310

[20] D. R. Brillinger, *Time Series, Data Analysis and Theory*, Expanded Edition, Holden Day Inc., 1981

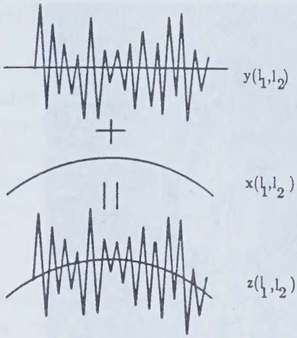

y(l_1,l_2)
+
x(l_1,l_2)
=
z(l_1,l_2)

Fig. 1: Superposition of a random and a deterministic functions.

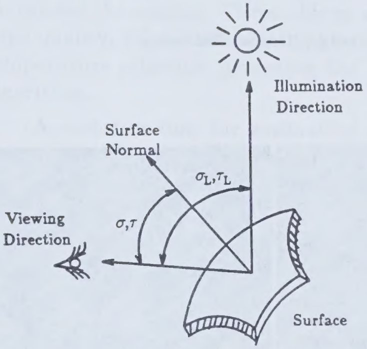

Illumination Direction
Surface Normal
σ_L, τ_L
Viewing Direction
σ, τ
Surface

Fig. 2: Three Different Directions on 3-D Surface

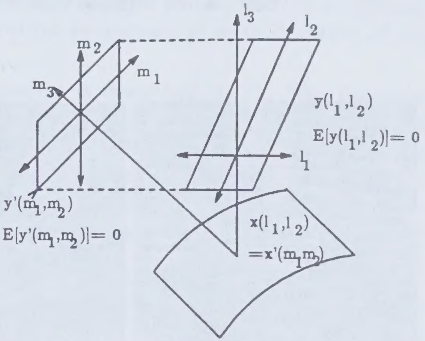
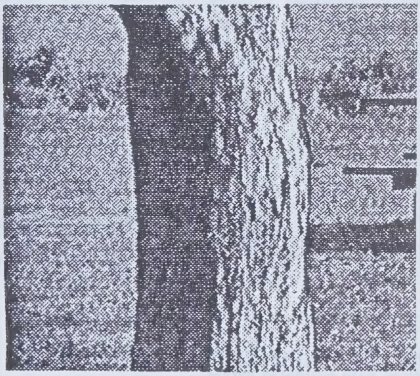
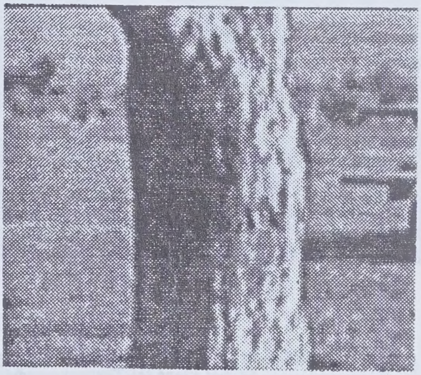
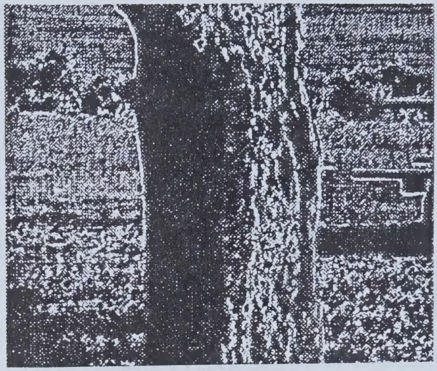

l_3
l_2
m_2
m_1
m_3
y(l_1,l_2)
E[y(l_1,l_2)]= 0
l_1
y'(m_1,m_2)
E[y'(m_1,m_2)]= 0
x(l_1,l_2)
=x'(m_1m_2)

Fig. 3: 3-D Geometry of the functions, x, x', y, y'.


(a)


(b)



(c)

Fig.4: A Tree Image; (a) Original (b) Smoothed (c) Random field image

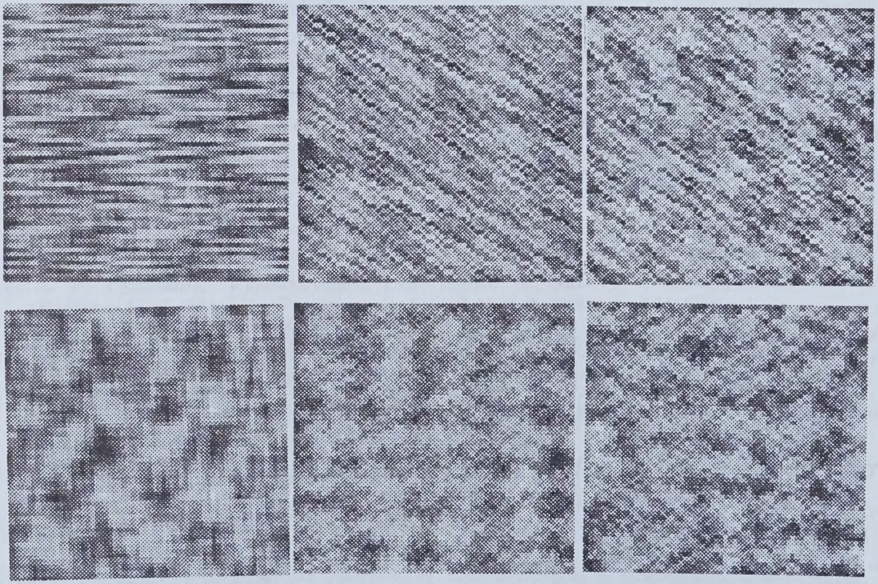


Fig.5: Synthesized Texture Images; (a) Original 2-D texture (b) Texture synthesized on 3-D Plane (c) Texture reconstructed with estimated values

EDGE DETECTION BY SIMULATED ANNEALING

H. L. TAN and E. J. DELP
Purdue University
Computer Vision and Image Processing Laboratory
School of Electrical Engineering
West Lafayette, Indiana 47907

1. INTRODUCTION

Simulated Annealing [1] is a stochastic optimization technique useful for finding relatively minimum cost solutions to NP complete combinatorial problems. The algorithm allows for "uphill climbs" while searching for the lowest cost solution. Edges in an image can be detected using a cost minimization approach based on Simulated Annealing. Three things are required: (1) a cost function for evaluating edge quality, (2) a method of generating new edge configurations, and (3) an efficient temperature schedule governing the temperature decrement at each iteration of the algorithm.

A cost function for evaluating edge quality is described in [2]. The function, denoted by $\mathbf{C}(S_m)$, assigns a cost value to each edge configuration S_m by considering the image data and the local structure of the edges. Assuming that the image has size $N \times N$, there is a total of 2^{N^2} possible edge configurations. The best configuration is the one with the minimum cost. Any exhaustive search technique is impractical because of the large number of configurations that would have to be considered.

2. GENERATING NEXT STATES

The second requirement of Simulated Annealing is a method of generating next states. Our method is based on a combination of five strategies described as follows. Assume that l is a given pixel site. The first strategy M_1 generates the next state by complementing a single pixel labeling at l in the present state. The second strategy M_2 complements two pixel labelings in the present state; both the pixels are constrained to lie in a 3×3 window region about l. The third strategy M_3 generates the next state by shifting the location of the edge pixel at l in the present state. The fourth strategy M_4 is similar to the third except that the shifting or perturbation of the edge location is more significant. The fifth strategy M_5 involves changing an arbitrary number of pixel labelings in a window region about l. The method of selecting l is to sequentially iterate through each pixel site in the image. This method of generating next states results in a corresponding Markov chain that is irreducible, having the property of weak reversibility as defined in [3].

3. TEMPERATURE VARIATION

Temperature, which acts as a control parameter at the kth iteration through the Simulated Annealing algorithm, is denoted by T_k. We used the following temperature schedule:

$$T_k = \frac{c}{\log(1+k)} .$$

The value of c is determined using Hajek's theorem [11]. A consequence of the theorem is that the annealing algorithm is guaranteed to converge to the set of globally minimum configurations when $c \geq d^*$, where d^* is the maximum depths of all the local minimums.

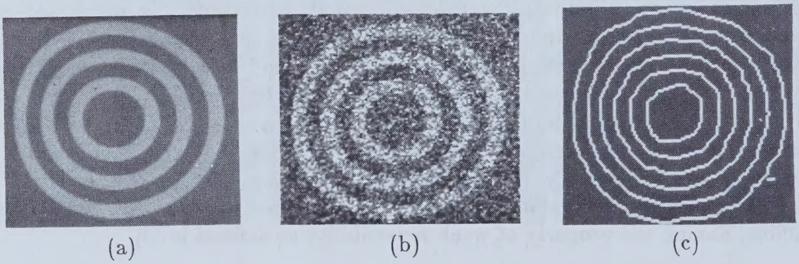
It is important to be able to find a good upper bound on the value of d^* since it determines how quickly we can decrease the temperature towards zero. This corresponds to how quickly we can drive the algorithm towards the global minimum of the cost function. Based on the method of generating next states in the algorithm, we estimate that

$$d^* < 2w_f + w_d - w_e .$$

We show an example of the edges detected using the Simulated Annealing approach described above. The image has size 128 × 128 and is made up of concentric rings of gray levels 115 and 140. It is constructed according to the specification in [4]. Figure 1(a) shows the rings image and Figure 1(b) shows the image corrupted with zero mean additive Gaussian noise of standard deviation 25. The detected edges are shown in Figure 1(c). Notice that all the edges are thin and continuous.

References

1. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, No. 4598, pp. 671-680, 13 May 1983.
2. H. L. Tan, "Edge detection by cost minimization," Ph.D. thesis, School of Electrical Engineering, Purdue Univ., West Lafayette, Indiana, Dec. 1988.
3. B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13 No.2, pp. 311-329, May 1988.
4. L. Kitchen and A. Rosenfeld, "Edge evaluation using local edge coherence," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-11, No. 9, pp. 597-605, IEEE, Sept. 1981.



(a) (b) (c)

Figure 1. Example of detected edges. (a) Rings image with gray levels 115 and 140. (b) Rings image corrupted with zero mean additive Gaussian noise of standard deviation 25. (c) Detected edges of noisy image.

STATISTICAL APPROACH TO NONDIFFRACTION COMPUTED TOMOGRAPHY IMAGING AND ITS APPLICATION IN IMAGE ANALYSIS [1]

TIANHU LEI and WILFRED SEWCHAND
School of Medicine and Medical System
University of Maryland
Baltimore, Maryland 21201

Transmission CT imaging, according to the physical nature of source-medium interaction, can be classified into two catogories, 1) a nondiffraction CT imaging, in which the interaction model and the external measurements (*i.e.*, projections) are characterized by the line integrals of some index of the medium (*i.e.*, object) and the image reconstruction is based on the Fourier Slice Theorem, 2) a diffraction CT imaging, in which the interaction and projections are modeled with the wave equation and the tomoraphic reconstruction approach is based on the Fourier Diffraction Theorem [1,2]. The former includes the conventional X-ray CT, emission CT, ultrasound CT (*e.g.*, refractive index CT and attenuation CT), and magnetic resonance imaging (MRI), *etc.*. The latter includes the acoustic, certain seismic, microwave, and optical imaging, *etc.*. This paper deals with the statistical analysis of the nondiffraction CT imaging and its application in image analysis.

Suppose that there are M groups of projections with $(2N + 1)$ projections in each group during the image reconstruction and the reconstructed image consists of J pixels. Then, for the most commonly used modes of CT data collection and the most commonly used convolution image reconstruction algorithm (which is equivalent to the Fourier Slice Theorem in the spatial frequency domain) [1,3], we proved that the image reconstruction is actually a transformation from $M(2N + 1)$ random variables (r.v.) to J r.v.s. More specifically, we proved [4] that nondiffraction CT image has the following global and local statistical properties:

1) The pixel image has an asymptotic normal distribution. Any two pixel images will be statistically independent, for a given probability value P, if the spatial separation between them is greater than N defined by

$$N = \begin{cases} \frac{1}{l_P} \cdot \frac{(2n_0+1)d}{\cos(\frac{\pi}{2}P)} & (\textit{parallel projection}) \\ \frac{1}{l_P} \cdot \frac{D\tan(2(n_0+n_0')+1)\delta}{\cos(\frac{\pi}{2}P)} & (\textit{divergent projections}) \end{cases} \tag{1}$$

where l_P is the pixel size; P is a specified probability value (usually quite large (say, $> 95\%$)); d and δ are the separations of the projections in the parallel and divergent beams, respectively; $(2n_0 + 1)$ is the effective bandwidth of the convolution function used in the image reconstruction algorithm; n_0' is the correlation interval corresponding to the α- dependent divergent projection and is equal to $[\frac{\alpha}{\delta}] + 1$ ($[x]$ denotes the integer part of x); D is the distance between the source position and the origin of the coordinate system.

2)If an image which is sampled from the original CT image with step size N satisfies (1) has isotropic property, then this sampled image is a (spatial) stationary process and ergodic in both mean and autocorrelation.

Based on the above statistical properties, a new stochastic model-based image segmentation technique has been developed for nondiffraction CT image. Its major steps are as follows.

A. The original image (IMG) is sampled with step size N of (1) to create the sampled

---

[1] The research work leading to this paper is supported by National Cancer Institute (NCI) under Grant #R01 CA 42180

images $SIMG_l$ $(l = 1, \cdots, L)$ (which is assumed to have I pixels)

$$IMG = \bigcup_{l=1}^{L} SIMG_l \tag{2}$$

B. $SIMG_l$ is modeled as a finite normal mixture [5], *i.e.*, its p.d.f. is

$$p(x) = \sum_{k=1}^{K_0} \pi_k p_k(x) \quad (0 < \pi_k < 1, \ \sum_{k=1}^{K_0} \pi_k = 1) \tag{3}$$

where $p_k(x)$ is the p.d.f. of the $k-th$ image class which is distributed as $N(\mu_k, \sigma_k)$.

C. The number of image classes in $SIMG_l$ is detected by an information criterion (AIC) [6]

$$AIC(K_0) = \min_{1 \le K \le K_{max}} AIC(K) = \min_{1 \le K \le K_{max}} \{-2\log(L(x_1, \cdots, x_I; \Theta(K))) + 2K_a\} \tag{4}$$

where $\Theta(K)$ is ML estimate of K parameters, $L(x_1, \cdots, x_I; \Theta(K))$ is the likelihood function of (3) evaluated at $\Theta(K)$, K_a is the number of independently adjustable parameters $K_a \le K$, and K_{max} is a prespecified value for K.

D. The parameters in (3) and (4) are estimated by a modified K-mean algorithm [7] which minimizes the modified Mahalanobis distance

$$(x_i - \mu_k)^T (\sigma_k^2)^{-1} (x_i - \mu_k) + \log \sigma_k^2 \tag{5}$$

E. Each $SIMG_l$ is segmented by using Bayesian decision criterion, which classifies pixel i into class k_0 if

$$L_{k_0}(x_i) = \max_{1 \le k \le K_0} L_k(x_i) \tag{6}$$

where $L_k(x_i)$ is the class-conditional likelihood function of $p_k(x)$.

F. Step C-E will be repeated L times so that the entire image IMG is segmented.

Results obtained by using simulated and real image data demonstrate the great promise of this technique.

References

[1] A.C.Kak and M.Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Press, New York, 1988.

[2] H.Stark (ed.)*Image Recovery: Theory and Application*, Academic Press, New York, 1987.

[3] G.T.Herman, *Image Reconstruction From Projections*, Academic Press, New York, 1980.

[4] T.Lei and W.Sewchand, (unpublished) 'Statistical approach to nondiffraction CT imaging'.

[5] D.M.Titterington, A.F.M.Smith, and U.E.Makov, *Statistical Analysis of Finite Mixture Distribution*, John Wiley & Sons, Chichester, 1985.

[6] M.Wax and T.Kailath, "Detection of signals by information theoretic criteria", *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP-33, Apr. 1985.

[7] J.T.Tou and R.C.Gonzalaz, *Pattern Recognition Principles*, Addison-Wesley Publication Company, Reading, MA., 1974.

SIMULTANEOUS IDENTIFICATION AND RESTORATION OF IMAGES USING MAXIMUM LIKELIHOOD ESTIMATION AND THE EM ALGORITHM[1]

K. T. LAY and A. K. KATSAGGELOS
Northwestern University
Department of Electrical Engineering and Computer Science
The Technological Institute
Evanston, IL 60208

In the restoration of a degraded image, the first step is the identification of the kind of degradation it has suffered, which usually needs to be based on the observed noisy-blurred image. Some recent work on image identification includes Tekalp et al.[1], Biemond et al.[2], Katsaggelos et al.[3] and Lagendijk et al.[4]. This paper focuses on simultaneous iterative blur and noise identification and restoration. The image and noise are modeled as multivariate Gaussian processes. Then we use maximum likelihood estimation (MLE) to estimate the parameters of the Gaussian process, where the estimation of the mean of the image represents the restored image.

It is assumed that the original image $x(i,j)$ and the observed image $y(i,j)$ are modeled as

$$x(i,j) = \sum_{(p,q)\in S_A} a(p,q)x(i-p,j-q) + u(i,j) \tag{1}$$

$$y(i,j) = \sum_{(p,q)\in S_D} d(p,q)x(i-p,j-q) + v(i,j) \tag{2}$$

where $u(i,j)$ and $v(i,j)$ are Gaussian noises, and the $d(i,j)'s$ specify the PSF. Rewriting the above equations in matrix form, by lexicographically ordering the image data, we have

$$x = Ax + u \tag{3}$$

$$y = Dx + v \tag{4}$$

The assumption that $(I - A)$ is invertible results in x being Gaussian distributed. Then the probability density function (pdf) of x and y are respectively

$$f_X(x) = |2\pi\Lambda_X|^{-1/2} exp\{\frac{-1}{2}(x-\mu_X)^T\Lambda_X^{-1}(x-\mu_X)\} \tag{5}$$

$$f_Y(y) = |2\pi(D\Lambda_X D^T + \Lambda_V)|^{-1/2} exp\{\frac{-1}{2}(y-D\mu_X)^T(D\Lambda_X D^T + \Lambda_V)^{-1}(y-D\mu_X)\} \tag{6}$$

We formulate the blur identification/image restoration problem as follows : "Given y, determine the unknowns μ_X,Λ_X,Λ_V and D, which maximize the likelihood function $f_Y(y)$." Then the resulting μ_X represents the restored image.

In order to emphasize that $f_Y(y)$ is parameterized, we write $f_Y(y) = f_Y(y;\theta)$, where $\theta = \{\mu_X, \Lambda_X, \Lambda_V, D\}$. The MLE of θ is the θ which maximizes $f_Y(y;\theta)$ (or equivalently, $\log f_Y(y;\theta)$), given the observed data y. Λ_V and Λ_X are usually parameterized by a small number of parameters (e.g. $\Lambda_V = \sigma_V^2 I$ when v is white). D is a block circulant matrix, determined by the $d(i,j)'s$ in S_D,whose size is usually small. As to μ_X, since it represents an image, which usually varies from pixel to pixel, it consists of a very large number of parameters. It would be too difficult to directly maximize $f_Y(y;\theta)$ because a large number of unknowns, dominated by μ_X, are involved. We propose that the set of unknown parameters θ be divided into two sets θ_1 and θ_2, where $\theta = \theta_1 \cup \theta_2$, $\theta_1 = \{\mu_X, \Lambda_X, \Lambda_V\}$ and $\theta_2 = \{D\}$. Given θ_2, θ_1 can be found with the aid of the EM algorithm, avoiding the difficulty of the too many unknowns in μ_X. Similarily, θ_2 can be obtained by directly maximizing $f_Y(y;\theta)$ while θ_1 is known.

---

[1] This material is based upon work supported in part by NSF under Grant No. MIP-8614217 and in part by NATO under Grant No. 0103/88.

The Expectation-Maximization(EM) algorithm[5,6] computes the MLE iteratively. Briefly speaking, the EM algorithm increases $\log f_Y(y)$ via the maximization (done in M-step) of the conditional expectation of $\log f_X(x)$ (obtained in E-step). More specifically, in the E-step,

$$Q(\phi|\phi^{(p)}) = E[\log f_X(x)|y;\phi^{(p)}] = \int_{\Omega_{X|y}} f_{X|Y}(x|y;\phi^{(p)}) \log f_X(x;\phi)\, dx \tag{7}$$

is computed where ϕ is the parameter specifying the relevant pdf's, and the integration is over all possible values of x that may produce the observed result y (denoted by $\Omega_{X|y}$). Then in the M-step, the parameter which maximizes $Q(\phi|\phi^{(p)})$ is chosen as the newly updated estimate(i.e. the best estimate so far). In a compact form, the EM algorithm is described by

$$\phi^{(p+1)} \longleftarrow max_{\{\phi\}} Q(\phi|\phi^{(p)}) \tag{8}$$

In applying the EM algorithm for estimating θ_1, the formulation is

$$\theta_1'' \longleftarrow max_{\{\theta_1\}} Q(\theta_1|\theta_1') \tag{9}$$

where for brevity we use θ_1'' for $\theta_1^{(p+1)}$ and θ_1' for $\theta_1^{(p)}$. In words, we make use of the current estimated parameter θ_1' to make a better estimation of the same parameter θ_1''. Then with this newly updated estimate as the current estimate, we proceed to find an even better one. The EM algorithm for the specific problem in consideration, under some symmetry conditions, can be shown to reduce to

$$\mu_X'' = \mu_X' + \Lambda_X' D^T (D\Lambda_X' D^T + \Lambda_V')^{-1}(y - D\mu_X') \tag{10}$$

$$\Lambda_X'' = \Lambda_X'(I - \Lambda_X' D^T (D\Lambda_X' D^T + \Lambda_V')^{-1} D\Lambda_X') \tag{11}$$

$$\Lambda_V'' = \Lambda_V'(I - \Lambda_V'(D\Lambda_X' D^T + \Lambda_V')^{-1}\Lambda_V') \tag{12}$$

According to Eq.(10), if $\mu_X' = 0$, the resulting restoration filter is a constrained least-squares filter. As to the estimation of θ_2(i.e. blur parameters), since the number of unknowns involved is not too large, $\log f_Y(y)$ can be directly maximized with respect to θ_2. This is equivalent to minimizing

$$J(D) = \log|D\Lambda_X D^T + \Lambda_V| + (y - D\mu_X)^T (D\Lambda_X D^T + \Lambda_V)^{-1}(y - D\mu_X) \tag{13}$$

In summary, the estimation of θ_1 and θ_2 is performed alternately. The key point is that the value of $\log f_Y(y)$ is increased from iteration to iteration. Further research tasks include : (a)the incorporation of constraints on $d(i,j)$ as well as $x(i,j)$ into the iterative process, as was done in [3]; (b) different assignment of the unknowns into the two sets θ_1 and θ_2.

References

[1] A. M. Tekalp, H. Kaufman and J. W. Woods, "Identification of Image and Blur Parameters for the Restoration of Noncausal Blurs", *IEEE Trans. on ASSP*, vol.34, No.4, pp.963-972, Aug. 1986.

[2] J. Biemond, F. G. van der Putten and J.W. Woods, "Identification and Restoration of Images with Symmetric Noncausal Blurs", *IEEE Trans. on CAS*, vol.35, no.4, pp.385-392, Apr. 1988.

[3] A. K. Katsaggelos, R. L. Lagendijk and J. Biemond, "Constrained Iterative Identification and Restoration of Images," *Proc. of EUSIPCO-88*, Grenoble, France, Sept. 1988.

[4] R. L. Lagendijk, D. L. Angwin, H. Kaufman and J. Biemond, "Recursive and Iterative Methods for Image Identification and Restoration," *Proc. of EUSIPCO-88*, Grenoble, France, pp.235-238, Sept. 1988.

[5] A. D. Dempster, N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM algorithm," *J. Roy. Stat. Soc.*, vol.B39, pp.1-37,1977.

[6] M. Feder and E. Weinstein, "Parameter Estimation of Superimposed Signals Using the EM algorithm" , *IEEE Tran. ASSP*, vol.36, no.4, pp.477-489, April 1988.

EFFICIENT AGREEMENT USING FAULT DIAGNOSIS

BRIAN A. COAN
Bell Communications Research
Morristown, NJ 07960–1910

Abstract: We give an extremely simple t-resilient (*i.e.,* tolerant of t processor faults) Byzantine agreement protocol that uses $t+1$ rounds of communication, $O(t^2)$ processors, small messages, and small local computation. This protocol uses roughly the same resources as a more complex protocol due to Dolev, Reischuk, and Strong. By adding fault diagnosis to our simple protocol, we produce a somewhat more complicated protocol that uses $t+1$ rounds, $O(t^{1.5})$ processors, small messages, and small local computation. Our work narrows the gap between achieved performance and the known lower bounds on resources needed by Byzantine agreement protocols.

1. INTRODUCTION

Reaching agreement is a fundamental problem in fault-tolerant distributed systems. It was first formalized by Pease, Shostak, and Lamport [12]. In deterministic synchronous systems subject to Byzantine processor faults there is a gap between the known lower bound on resources and the best performance of a known protocol. We narrow that gap. Specifically we reduce the number of processors needed by an agreement protocol that uses "small" messages and terminates in an optimal number of rounds.

We are interested in minimizing the use of the following three resources: processors, rounds of communication, and message size. For message size we limit our interest to distinguishing between polynomial and exponential complexity. If t is an upper bound on the number of processor faults then an optimal protocol uses $3t+1$ processors [8, 12], $t+1$ rounds [7], and messages whose size is polynomial in t. At present there is a trade-off in the use of these resources. Protocols are know that are optimal in any two of the measures, but no protocol is optimal in all three. The original agreement protocol of Lamport, Shostak, and Pease [10] uses $3t+1$ processors, $t+1$ rounds, and messages whose size is exponential in t. Many protocols [2, 3, 4, 6, 9, 14, 15] use $3t+1$ processors, more than $t+1$ rounds, and messages whose size is polynomial in t. A protocol of Dolev, Reischuk, and Strong [5] uses $2t^2+O(t)$ processors, $t+1$ rounds, and messages whose size is polynomial in t.

We give two new protocols. The first is an extremely simple protocol that uses roughly the same resources as the more complex protocol of Dolev, Reischuk, and Strong. Our protocol has the sort of implicit fault diagnosis that was first used by Reischuk [13]. The processors are partitioned into $t+1$ disjoint groups of size $4t+1$. Each round a different group plays a special role. We call this group the *dominant* group. The key property of the protocol is that the decision of all correct processors is "committed" by the end of the first round in which the dominant group contains only correct processors. A decision is eventually "committed" because there is at least one group that consists entirely of correct processors. This protocol has implicit fault diagnosis in that the failure to "commit" a decision by a round implies the existence of a faulty processor in the dominant group for that round. A similar protocol was developed independently by Bar-Noy [1].

Our second protocol is constructed by adding explicit fault diagnosis to the simple protocol. The explicit fault-diagnosis scheme is similar to one used by Dolev, Reischuk, and Strong [5]. Our objective is to reduce the number of processors by "reusing" groups (*i.e.,* having the same group dominant in more than one round). In most rounds we do fault diagnosis on all of the processors in the previous round's dominant group. Often fault diagnosis allows each correct processor to correctly identify a faulty processor. A correct processor ignores certain messages from processors that it has diagnosed as faulty. After using a group as the dominant group and doing fault diagnosis on the group, we can productively reuse the group as a dominant group in a subsequent round. The decision of the correct processors will be "committed" by the first round in which the dominant group contains no undiagnosed faulty processors; this happens by round $t+1$. We reuse each group $O(\sqrt{t})$ times in an execution. This reduces the number of groups to $O(\sqrt{t})$, and reduces the total number of processors to $4t^{1.5}+O(t)$.

In our protocols correct processors send small messages. In the first protocol messages are of size $O(\log |V|)$, where V is the set of possible inputs; in the second protocol messages are of size $O(t \cdot \log |V|)$. In each protocol a correct processor does an amount of computation that is linear in the number of message bits that it receives.

The worst-case lower bound on rounds to reach agreement is $t+1$. Dolev, Reischuk, and Strong [5] pointed out that it is possible for a protocol to terminate faster in some executions. They gave the name *early stopping* to this property. Our protocols have early stopping. In an execution where f processors actually fail, our protocols terminate in $\min(t+1, f+2)$ rounds, which is optimal [5]. As with any early stopping protocol which has a known upper bound on rounds, simple variants of our protocols have simultaneous termination (all correct processors decide in the same round). It is known that it is impossible to have both simultaneous termination and early stopping [5].

In Section 2 we review the model. In Section 3 we give the simple protocol and in Section 4 we give the protocol with explicit fault diagnosis. In Section 5 we mention an interesting new protocol of Moses and Waarts [11].

2. MODEL AND PROBLEM STATEMENT

We consider computer systems that consist of a collection of processors which interact by sending messages over links. Each link provides a bidirectional communication path between a pair of processors. The collection of links constitutes the communication network. It is known [8] that no agreement protocol is possible unless the connectivity of the communication network is high. Throughout this paper we assume that the communication network is fully connected and reliable.

Our timing model embodies the set of assumptions that we make regarding bounds on the relative rates of processors and bounds on message delivery time. We assume a *synchronous* timing model in which all processors act in unison. An execution of a protocol consists of a series of *rounds*. In each round each processor sends messages, receives all of the messages sent to it, and acts on the messages received.

In an execution some processors are *correct*. The rest are *faulty*. A correct processor sends messages according to its protocol. A faulty processor can deviate from its protocol. We are interested in protocols that tolerate *Byzantine* processor faults. In the Byzantine fault model there are no constraints on the contents of messages sent by a faulty processor; however, the recipient of any message knows the identity of the sender. Throughout this paper we let t be a fixed upper bound on the number of faulty processors and we let n denote the total number of processors. For the Byzantine fault model it is known [8, 10] that there are no agreement protocols unless $n \geq 3t+1$.

In a protocol that solves the agreement problem each processor begins with an input from some fixed finite set V. Without loss of generality, we assume that V is totally ordered, $0 \in V$, and $\perp \notin V$. We refer to the elements of the set V as *values*. Each correct processor may, at any point during an execution of a protocol, irrevocably decide on a value (element of V) as its answer. There are three conditions that the correct processors must satisfy in any execution.

- *Agreement condition:* All correct processors that decide reach the same decision.
- *Validity condition:* If all correct processors start with input $v \in V$, then v is the decision of all of the correct processors that decide.
- *Termination condition:* All correct processors eventually decide.

In calculating the number of message bits sent and the number of local computation steps used by a protocol, we follow the standard practice of only counting resources used by correct processors. This is necessary because there is no *a priori* bound on the resources used by a faulty processor. It is reasonable to focus on the resources used by correct processors because they are presumably the only processors doing productive work.

3. A SIMPLE PROTOCOL

Let $n = 4t^2 + 5t + 1$. We use the set $\{1, \ldots, n\}$ to index the processors. Partition the processors into $t+1$ disjoint *groups* each consisting of $4t+1$ processors. Call the groups $G_1, \ldots, G_{t+1}$. For each $r \in \{1, \ldots, t+1\}$, the group G_r is *r-dominant*. Each processor in an r-dominant group is *r-dominant*. The intuition is that the r-dominant processors play a special role in round r of our protocol.

Processor groups are important in our analysis of the running time of Protocol 1. Suppose $f \leq t$ is the actual number of faulty processors in an execution. Because there are $t+1$ groups, it follows that, for some $r \leq f+1$, there is an r-dominant group consisting entirely of correct processors. We will show that all correct processors decide by round $\min(t+1, r+1)$ in any execution where the r-dominant group consists entirely of correct processors. We conclude that all correct processors decide by round $\min(t+1, f+2)$.

In Protocol 1 the variable FAVOR holds the value that a correct processor currently *favors* as its decision. Each correct processor initializes FAVOR to be its input value. In each round each correct processor selects a new favored value and possibly decides. The following specific steps occur in round r. In step 1 each r-dominant correct processor broadcasts (*i.e.,* sends to all processors including itself) its favored value. In step 2 a correct processor receives the favored value of each r-dominant processor, substituting 0 for any value that is missing or invalid (*i.e.,* not an element of V). This substitution is never necessary on a message received from a correct processor. In step 3 each correct processor chooses a new favored value. The value chosen is the most frequent value received in step 2 (after substitution). If several values are tied then the smallest of the tied values is chosen. In step 4 a correct undecided processor may decide. It decides on its favored value if either $r = t+1$ or it observes that $3t+1$ or more r-dominant processors sent round r messages favoring that value.

```
    Initialization for processor $p$:
        Let FAVOR be the input to processor $p$.
    Code for processor $p$ in round $r$, for all $r \in \{1, \ldots, t+1\}$:
1.      If $p \in G_r$ then broadcast FAVOR.
2.      For each $q \in G_r$
            Receive the message $\mathrm{RAW}_q$ from processor $q$.
            If $\mathrm{RAW}_q \in V$ then let $\mathrm{MSG}_q = \mathrm{RAW}_q$ else let $\mathrm{MSG}_q = 0$.
3.      Let FAVOR be the most frequent element of the multiset $\{\mathrm{MSG}_q : q \in G_r\}$,
            breaking ties by choosing the smallest of the tied elements.
4.      If processor $p$ is still undecided then
            If $r = t+1$ or $|\{q \in G_r : \mathrm{MSG}_q = \mathrm{FAVOR}\}| \geq 3t+1$ then decide FAVOR.
```

Protocol 1: A Simple Protocol

We say that correct processor p has an *r-vote* for value v from processor q if processor q is r-dominant and processor p has $\mathrm{MSG}_q = v$ at the end of step 2 of round r.

Lemma 1: *Suppose that $r \in \{1, \ldots, t+1\}$ and p is a correct processor that is undecided at the start of round r. If each correct processor has FAVOR $= v$ at the end of round r then processor p decides v by round $\min(t+1, r+1)$.*

Proof: If $r = t+1$ then processor p decides v in step 4 of round r because FAVOR $= v$ at the end of round r at processor p. Suppose instead that $r \leq t$. Either processor p decides in round r or it does not. If processor p decides in round r then it decides v. If processor p does not decide in round r then it has an $(r+1)$-vote for v from each $(r+1)$-dominant correct processor. This is $3t+1$ or more $(r+1)$-votes for v among the $4t+1$ total number of $(r+1)$-votes seen by processor p. Thus processor p decides v in step 4 of round $r+1$. In either event processor p decides v by round $r+1 \leq t+1$. □

Lemma 2: *If there are $f \leq t$ faulty processors then there is a round $d \leq f+1$ and a value v such that all correct processors end round d with FAVOR $= v$.*

Proof: Let d be the smallest integer such that all d-dominant processors are correct. Because there are at least $f+1$ disjoint processor groups and at most f faulty processors, d exists and $d \leq f+1$. Because all d-dominant processors are correct, each correct processor has the same multiset of d-votes. Because all correct processors apply precisely the same selection rule to precisely the same multiset of d-votes in step 3 of round d, there is some v such that all correct processors end round d with FAVOR $= v$. □

Theorem 3: *Protocol 1 solves the agreement problem.*

Proof: We show that the agreement, validity, and termination conditions are satisfied.

Agreement condition: Step 4 of the code ensures that all correct processors decide by round $t+1$. Let r be the first round in which a correct processor decides. There are two cases. The first case is $r \leq t$. Suppose that processor p decides v in round r. Step 4 ensures that processor p has $3t+1$ or more r-votes for v. At least $2t+1$ of these r-votes are from correct processors. Thus all correct processors have $2t+1$ or more r-votes for v and at most $2t$ other r-votes. It follows that all correct processors end round r with $\text{FAVOR} = v$. By the choice of r, all correct processors are undecided at the start of round r. By Lemma 1 all correct processors decide v. The second case is $r = t+1$. By Lemma 2 there is some round $d \leq t+1$ and some value v such that all correct processors end round d with $\text{FAVOR} = v$. By the choice of r, all correct processors are undecided at the start of round r. Because $d \leq r$ all correct processors are undecided at the start of round d. By Lemma 1 all correct processors decide v.

Validity condition: Let v be the common input of all of the correct processors. All correct processors begin round 1 with $\text{FAVOR} = v$. Each correct processor has $3t+1$ or more 1-votes for v and at most t other 1-votes. Thus each correct processor sets FAVOR to v in step 3 and decides v in step 4 of round 1.

Termination condition: Step 4 of the code ensures that all correct processors decide by round $t+1$. □

Theorem 4: *If there are $f \leq t$ faulty processors then Protocol 1 uses the following resources:* $\min(t+1, f+2)$ *rounds,* $4t^2+5t+1$ *processors,* $O(t^4 \cdot \log|V|)$ *message bits, and* $O(t^4 \cdot \log|V|)$ *local computation steps.*

Proof: We begin with the bound on rounds. By Lemma 2 there is a round $d \leq f+1$ and a value v such that all correct processors end round d with $\text{FAVOR} = v$. By Lemma 1 all correct processors decide by round $\min(t+1, f+2)$.

The bound on processors is immediate from the statement of the protocol. In each of $t+1$ rounds a group of $O(t)$ correct processors sends messages of size $\log|V|$ to each of the $O(t^2)$ processors. Thus a total of $O(t^4 \cdot \log|V|)$ message bits are sent. The bound on local computation follows because a correct processor in Protocol 1 does a constant amount of work for each message bit. □

As we have written our protocol, messages are sent in each of $t+1$ rounds. This could be improved. A correct processor can stop sending two rounds after it decides. This is because all other correct processors have decided by then. We omit a detailed proof.

It is possible to modify Protocol 1 to reduce the number of processors by a factor of two. This modification consists (primarily) of reducing the size of some of the groups. In Protocol 1 the group size was chosen based on the worst-case assumption that all of the faulty processors might be in a single group. We can do a tighter analysis based on the observation that the size of group G is relevant only in those execution where some correct processor is undecided in the round when group G is dominant. We can reduce the size of those groups with a large index (*i.e.,* groups that are dominant in later rounds), because these groups are reached only if there are a sufficient number of faulty processors in the earlier groups. We omit further details.

4. A PROTOCOL WITH EXPLICIT FAULT DIAGNOSIS

Protocol 1 has implicit fault diagnosis. For each round (after round 2) that the correct processors fail to decide, we can deduce that a particular group contains a faulty processor. Protocol 2 is a variant of Protocol 1 with explicit fault diagnosis. This explicit fault diagnosis allows "reuse" of processor groups and thereby reduces the total number of processors from $4t^2 + O(t)$ to $4t^{1.5} + O(t)$. Specifically, for any group with at most $\lfloor\sqrt{t}\rfloor$ faulty processors, we will show that each round the group is dominant results in one of the following. Either (1) all correct processors have decided by the following round or (2) each correct processor is able to identify a (previously unidentified) faulty processor. Each group is dominant for $\lfloor\sqrt{t}+1\rfloor$ consecutive rounds. Eventually there are no more faulty processors to identify (*i.e.,* option 2 is impossible). Then all processors decide in the following round (option 1).

We require $\lceil\sqrt{t}\rceil$ groups to assure that there will be at least one group which contains at most $\lfloor\sqrt{t}\rfloor$ faulty processors. There may be many groups that contain more than $\lfloor\sqrt{t}\rfloor$ faulty processors. In these groups our fault diagnosis scheme can fail, falsely identifying correct processors as faulty. To cope with these erroneous diagnoses, we increase the group size from $4t+1$ to $\lfloor 4t+\sqrt{t}+1\rfloor$ and we disregard the results of fault diagnosis if more than $\lfloor\sqrt{t}\rfloor$ processors are diagnosed as faulty. This is enough to ensure that nothing bad (*e.g.*, disagreement) happens when a group with more than $\lfloor\sqrt{t}\rfloor$ faulty processors is dominant.

An execution is divided into *blocks* of $\lfloor\sqrt{t}+1\rfloor$ consecutive rounds. The blocks are numbered starting with block 1. We subdivide a block into *phases,* each one round long. The first phase in a block is phase 1 and the last is phase $\lfloor\sqrt{t}+1\rfloor$. We define the functions block and phase. Let $r \in \{1,\ldots,t+1\}$ be a round.

- $\text{block}(r) = \lceil r/\lfloor\sqrt{t}+1\rfloor\rceil$ is the block of which round r is a part.
- $\text{phase}(r) = r - (\text{block}(r) - 1) \cdot \lfloor\sqrt{t}+1\rfloor$ is the phase of round r.

For Protocol 2 we let $n = \lceil\sqrt{t}\rceil \cdot \lfloor 4t+\sqrt{t}+1\rfloor$. We use the set $\{1,\ldots,n\}$ to index the processors. Partition the processors into $\lceil\sqrt{t}\rceil$ disjoint *groups* each consisting of $\lfloor 4t+\sqrt{t}+1\rfloor$ consecutively numbered processors. Call these groups $G_1,\ldots,G_{\lceil\sqrt{t}\rceil}$. Group G is *good* if it contains at most $\lfloor\sqrt{t}\rfloor$ faulty processors; otherwise, it is *bad.* For each $r \geq 1$ the group G_b is *r-dominant* if $b = \text{block}(r)$. Each processor in an r-dominant group is *r-dominant.* We sometimes find it convenient to write that an r-dominant processor or group is "dominant in round r."

In each round of Protocol 2, each correct processor engages in the following activities: local computation (step 1), fault diagnosis (steps 2–8), selection of the value that it currently favors as its decision (steps 9–12), and possible selection of a decision (step 13). Note the strong similarity between Protocol 1 and steps 9–13 of Protocol 2. A more detailed description of Protocol 2 follows.

In Protocol 2 the variable FAVOR holds the value that a correct processor currently *favors* as its decision. Each correct processor initializes FAVOR to be its input value. The following specific steps occur in round r. In step 1 a correct processor sets b to the current block number, l to the lowest index of a b-dominant processor, and h to the highest index of a b-dominant processor. A correct processor may do fault diagnosis on the b-dominant processors in steps 2–8. After step 8 the set FAULTY contains the indices of those processors (if any) that have been diagnosed as faulty. If $\text{phase}(r) = 1$ then a correct processor does no fault diagnosis and it sets FAULTY to be the empty set in step 3. If $\text{phase}(r) \geq 2$ then each correct processor does fault diagnosis in steps 5–8. In step 5 each b-dominant processor broadcasts the vector $(\text{MSG}_l,\ldots,\text{MSG}_h)$. A $\perp$ entry in this vector indicates that the corresponding processor was diagnosed as faulty in round $r-1$. Any other entry is the message received from the corresponding processor in step 10 of round $r-1$. In steps 6–8 a processor p is diagnosed as correct if there is a value v and a sufficiently large group of processors each of which reports that (1) it got the message v from processor p in step 10 of round $r-1$ and (2) it did not diagnose processor p as faulty in round $r-1$; otherwise, processor p is diagnosed as faulty.

After fault diagnosis is concluded, the following steps are performed in round r. In step 9 each b-dominant correct processor broadcasts its favored value. In step 10 a correct processor receives the favored value of each b-dominant processor, substituting 0 for any favored value that is missing or invalid (*i.e.*, not an element of V). This substitution is never necessary on a message received from a correct processor. In step 11 a correct processor may "ignore" (*i.e.*, set to $\perp$) messages from processors that it has diagnosed as faulty. "Ignoring" messages is done only if there are $\lfloor\sqrt{t}\rfloor$ or fewer processors in FAULTY. In step 12 each correct processor chooses a new favored value. The value chosen is the most frequent (non-$\perp$) value received in step 10 (after substitution and ignoring). If several values are tied then the smallest of the tied values is chosen. In step 13 a correct undecided processor may decide. It decides on its favored value if either $r = t+1$ or it observes that $\lfloor 3t+\sqrt{t}+1\rfloor$ or more b-dominant processors sent round r messages favoring that value.

In our model each round consists of sending followed by receiving followed by changing state. In the code as written, sending and receiving are interspersed within each round. This is done to make the code easy to read. Because there is no dataflow from the receipt of messages to the sending of messages (within a round), it is straightforward to translate Protocol 2 into the standard form. We omit this translation.

```
Initialization for processor $p$:
    Let $\textsc{favor}$ be the input to processor $p$.
Code for processor $p$ in round $r$, for all $r \in \{1, \ldots, t+1\}$:
1.      Let $b = \text{block}(r)$, let $l = \min(G_b)$, and let $h = \max(G_b)$.
2.      If phase$(r) = 1$ then
3.          Let $\textsc{faulty} = \emptyset$.
4.      else
5.          If $p \in G_b$ then broadcast $(\textsc{msg}_l, \ldots, \textsc{msg}_h)$.
6.          For each $q, u \in G_b$
                Receive the message $(\textsc{raw}_{q,l}, \ldots, \textsc{raw}_{q,h})$ from processor $q$.
                If $\textsc{raw}_{q,u} \in V \cup \{\bot\}$ then let $\textsc{echo}_{q,u} = \textsc{raw}_{q,u}$ else let $\textsc{echo}_{q,u} = \bot$.
7.          For each $q \in G_b$
                Let $\textsc{count}_q$ be the number of occurrences of the most frequent
                non-$\bot$ element in the multiset $\{\textsc{echo}_{u,q} : u \in G_b\}$.
8.          Let $\textsc{faulty} = \{q \in G_b : \textsc{count}_q < 4t+1\}$.
9.      If $p \in G_b$ then broadcast $\textsc{favor}$.
10.     For each $q \in G_b$
            Receive the message $\textsc{raw}_q$ from processor $q$.
            If $\textsc{raw}_q \in V$ then let $\textsc{msg}_q = \textsc{raw}_q$ else let $\textsc{msg}_q = 0$.
11.     If $|\textsc{faulty}| \leq \lfloor\sqrt{t}\rfloor$ then for each $q \in \textsc{faulty}$ let $\textsc{msg}_q = \bot$.
12.     Let $\textsc{favor}$ be the most frequent non-$\bot$ element of the multiset $\{\textsc{msg}_q : q \in G_b\}$,
            breaking ties by choosing the smallest of the tied elements.
13.     If processor $p$ is still undecided then
            If $r = t+1$ or $|\{q \in G_b : \textsc{raw}_q = \textsc{favor}\}| \geq \lfloor 3t + \sqrt{t} + 1\rfloor$ then
                decide $\textsc{favor}$.
```

Protocol 2: A Protocol with Fault Diagnosis

We make several definitions for use in our proof. Correct processor p has a *tentative r-vote* (resp., *final r-vote*) from processor q for $v \in V$ (resp., $v \in V \cup \{\bot\}$) if processor q is r-dominant and processor p has $\textsc{msg}_q = v$ at the end of step 10 (resp., step 11) of round r. Round $r \in \{1, \ldots, t\}$ is *decisive* if there is a set of at least $\lfloor 2t + \sqrt{t} + 1\rfloor$ correct $(r+1)$-dominant processors which all have the same multiset of final r-votes. Decisive and non-decisive rounds play a fundamental role in our proof of Protocol 2. Each non-decisive round results in the diagnosis of a faulty processor. If the dominant group is bad then the diagnosis is implicit (*i.e.,* similar to what happens in Protocol 1). If the dominant group is good, then the diagnosis is explicit (*i.e.,* there is a specific faulty processor that is identified by all of the correct processors). There are a limited number of faulty processors to diagnose. Eventually there will be either a decisive round or a round at the start of which all faulty processors have been diagnosed. All correct processors decide shortly after the first such round (at the latest).

For all correct processors p and rounds r, we let $F_{p,r}$ be the value of the variable $\textsc{faulty}$ at processor p at the end of step 8 of round r. These sets contain the results of the explicit fault diagnosis procedure. If the dominant group is bad, explicit fault diagnosis fails and the content of these sets is unpredictable. If the dominant group is good then these sets contain only faulty processors. We let F_r be the intersection over all correct processors p of $F_{p,r}$. Thus a processor is in F_r if and only if it has been diagnosed as faulty by all correct processors.

For all processors p and rounds r, if there is a $v \in V$ such that at most $\lfloor\sqrt{t}\rfloor$ correct r-dominant processors fail to have have a final r-vote for v from processor p then we let $V_{p,r} = v$; otherwise we let $V_{p,r} = \bot$. (Recall that a final vote is an element of $V \cup \{\bot\}$.) A final r-vote from processor p is *common* if it is for $V_{p,r}$; it is *rare* otherwise. The concept of rare final votes provides the link between non-decisive rounds and fault diagnosis. In Lemma 5 we show that many processors get rare final votes in any non-decisive round. In Lemma 8 we use this characterization of a non-decisive round to show that some specific faulty processor is diagnosed for each non-decisive round that a good group is dominant.

Lemma 5: *Suppose that $r \in \{1, \ldots, t\}$. If round r is not decisive then there is a set of $t+1$ correct*

$(r+1)$-dominant processors, each with at least one rare final r-vote.

Proof: We prove the contrapositive. The $(r+1)$-dominant group consists of $\lfloor 4t+\sqrt{t}+1\rfloor$ processors, at most t of which are faulty. It is immediate that there are at least $\lfloor 3t+\sqrt{t}+1\rfloor$ correct $(r+1)$-dominant processors. By assumption, at most t correct $(r+1)$-dominant processors have any rare final r-votes. Thus there are at least $\lfloor 2t+\sqrt{t}+1\rfloor$ correct $(r+1)$-dominant processors which only have common final r-votes. All of these processors have the same multiset of final r-votes. By definition, round r is decisive. □

In Lemma 6 we show that, when a good group is dominant, our fault-diagnosis scheme never erroneously diagnoses a correct processor as faulty.

Lemma 6: *Suppose that $r \in \{1,\ldots,t+1\}$. If the r-dominant group is good, p is any correct processor, and $q \in F_{p,r}$ then q is a faulty r-dominant processor.*

Proof: The proof is by induction on phase(r).

Basis: (phase(r) = 1) It is clear from the code that $F_{p,r} = \emptyset$. The claim is vacuously true because there are no processors in the set $F_{p,r}$.

Induction: Because phase(r) ≥ 2, the r-dominant group and the $(r-1)$-dominant group are the same group. If there is a correct processor u with $q \in F_{u,r-1}$ then q is a faulty r-dominant processor by the induction hypothesis. Suppose instead that $q \notin F_{u,r-1}$ for all correct processors u. Processor p places processor q in the set FAULTY in step 8 of round r. It is clear from step 8 that processor q is r-dominant. We now show that processor q is faulty. Suppose to the contrary that processor q is correct. Let v be the value of FAVOR at processor q at the start of round $r-1$. Since processor q is correct and r-dominant, each correct processor has a tentative $(r-1)$-vote for v from processor q. Since $q \notin F_{u,r-1}$ for all correct processors u, each correct processor has a final $(r-1)$-vote for v from processor q. In step 5 of round r each correct r-dominant processor (at least $4t+1$ processors because a good group is r-dominant) reports to processor p that it has a final $(r-1)$-vote for v from processor q. In step 7 processor p counts $4t+1$ or more r-dominant processors that report having a final $(r-1)$-vote for v from processor q. Therefore processor p does not include processor q in the set FAULTY in step 8 of round r and $q \notin F_{p,r}$. This contradicts the assumption that $q \in F_{p,r}$; therefore, processor q must be a faulty r-dominant processor. □

In Lemma 7 we show that, when a good group is dominant, rare votes come only from faulty dominant processors.

Lemma 7: *Suppose that $r \in \{1,\ldots,t+1\}$. If the r-dominant group is good and some correct processor has a rare final r-vote from processor q then processor q is a faulty r-dominant processor.*

Proof: It follows immediately from the definition of final votes that processor q is r-dominant. We now show that processor q is faulty. Suppose to the contrary that processor q is correct. Let v by the value of FAVOR at processor q at the start of round r. Since processor q is correct and r-dominant, each correct processor has a tentative r-vote for v from processor q. We have assumed that the r-dominant group is good and that q is a correct processor. By Lemma 6, there is no correct processor p such that $q \in F_{p,r}$. Thus each correct processor has a final r-vote for v from processor q. It follows easily that $V_{q,r} = v$ and each correct processor has a common final r-vote from processor q. This contradicts the assumption that some correct processor has a rare final r-vote from processor q; therefore, processor q must be a faulty r-dominant processor. □

Lemma 8 provides the following characterization of our fault-diagnosis scheme: some specific faulty processor is diagnosed for each non-decisive round that a good group is dominant.

Lemma 8: *Suppose that $r \in \{1,\ldots,t+1\}$. If the r-dominant group is good and there is no decisive round in the interval from round $r -$ phase(r) $+ 1$ through round $r-1$ then $|F_r| \geq$ phase(r) $- 1$.*

Proof: The proof is by induction on phase(r).

Basis: (phase(r) = 1) It is immediate that $|F_r| \geq 0 =$ phase(r) $- 1$.

Induction: Let $k =$ phase(r) $- 1$, let G be the set of r-dominant processors, and let $B = \{q \in G : V_{q,r-1} = \bot\}$. First we show that $|B| \geq k$. Then we show that $B \subset F_r$. This implies that $|F_r| \geq k$.

By Lemma 6 and the assumption that group G is good, $|F_{p,r-1}| \le \lfloor\sqrt{t}\rfloor$ for all correct processors p. Thus step 11 ensures that $\perp$ is the final $(r-1)$-vote that an arbitrary correct processor has from an arbitrary member of F_{r-1}. It follows immediately that $F_{r-1} \subset B$ and no correct processor has a rare final $(r-1)$-vote from any processor in F_{r-1}. By the induction hypothesis $|F_{r-1}| \ge k-1$. We now identify a processor in the set $B - F_{r-1}$ and therefore conclude that $|B| \ge k$. By assumption round $r-1$ is not decisive. By Lemma 5 there is a set W of $t+1$ correct $(r-1)$-dominant processors, each with at least one rare final $(r-1)$-vote. By Lemma 7, these rare votes all come from faulty $(r-1)$-dominant processors. By assumption there are at most $\lfloor\sqrt{t}\rfloor$ faulty $(r-1)$-dominant processors. By the pigeonhole principal there must be some processor q responsible for at least $\lfloor\sqrt{t}+1\rfloor$ rare votes. $\text{COUNT}_q \le 4t$ at any correct processor in step 8 of round r. It follows easily that $V_{q,r-1} = \perp$ and $q \in B$. We have $q \notin F_{r-1}$ because no correct processor has a rare final $(r-1)$-vote from any processor in F_{r-1}. Thus $|B| \ge k$.

Let s be any processor in the set B and let p be any correct processor. By the definition of the set B, $V_{s,r-1} = \perp$. It follows that $\text{COUNT}_s \le 4t$ in step 8 of round r at processor p. From step 8 of the code and the definition of $F_{p,r}$, we have $s \in F_{p,r}$. Because p is an arbitrarily chosen correct processor, $s \in F_r$. Because s is an arbitrarily chosen element of B, we have $B \subset F_r$. □

In Lemma 9 we establish a key relationship between the number of tentative votes a processor has for a value and the number of final votes it has for that value.

Lemma 9: *Suppose that $r \in \{1,\ldots,t+1\}$ and $v \in V$. If correct processor p has x tentative r-votes for v and x' final r-votes for v then $\lfloor x - \sqrt{t}\rfloor \le x' \le x$.*

Proof: Let G be the r-dominant group. From the definition of tentative r-votes we have that $x = |\{q \in G : M_q = v\}|$, where M is the value of the array MSG at processor p at the end of step 10 of round r. In step 11 processor p changes at most $\lfloor\sqrt{t}\rfloor$ elements of its array MSG to $\perp$. We have $\perp \ne v$ because $\perp \notin V$ and $v \in V$. By the definition of final r-votes, $x' = |\{q \in G : M'_q = v\}|$, where M' is the value of the array MSG at processor p at the end of step 11 of round r. Thus $\lfloor x - \sqrt{t}\rfloor \le x' \le x$. □

Lemmas 10 and 11 are analogous to Lemmas 1 and 2, which we proved for Protocol 1. The proof of Lemma 10 is similar to the proof of Lemma 1. The proof of Lemma 11 is quite different from (and more complex than) the proof of Lemma 2.

Lemma 10: *Suppose that $r \in \{1,\ldots,t+1\}$ and p is a correct processor that is undecided at the start of round r. If each correct processor has FAVOR $= v$ at the end of round r then processor p decides v by round $\min(t+1, r+1)$.*

Proof: If $r = t+1$ then processor p decides v in step 13 of round r, because FAVOR $= v$ at the end of round r at processor p. Suppose instead that $r \le t$. Either processor p decides in round r or it does not. If processor p decides in round r then it decides v. If processor p does not decide in round r then it has a tentative $(r+1)$-vote for v from each $(r+1)$-dominant correct processor. So it has at least $\lfloor 3t + \sqrt{t} + 1\rfloor$ tentative $(r+1)$-votes for v and at most t tentative $(r+1)$-votes for any $v' \ne v$. Thus processor p decides v in step 13 of round $r+1$. In either event processor p decides v by round $r+1 \le t+1$. □

Lemma 11: *If there are $f \le t$ faulty processors then there is a round $d \le f+1$ and a value v such that all correct processors end round d with FAVOR $= v$.*

Proof: Let b be the number of faulty processors in bad groups and let w be the number of rounds in which the dominant group is bad. Clearly $b \le t$. We have $w \le b$ because each bad group contains at least $\lfloor\sqrt{t}+1\rfloor$ faulty processors and is dominant in at most $\lfloor\sqrt{t}+1\rfloor$ rounds. Thus $w \le t$ and there must be at least one round in which a good group is dominant. Let G be the first good group that is dominant, let r be the first round in which group G is dominant, and let f' be the number of faulty processors in group G. We have $r \le w+1$ because w is an upper bound on the number of consecutive rounds with a bad group dominant. Because faulty processors in good groups are disjoint from faulty processors in group G, we get $b + f' \le f$. Starting at round r, group G is dominant until either $\lfloor\sqrt{t}+1\rfloor$ rounds have elapsed or the end of round $t+1$, whichever comes first. Group G is dominant from round r through at least round $r + f'$ because $f' + 1 \le \lfloor\sqrt{t}+1\rfloor$ and $r + f' \le t+1$. Let c be the number of consecutive non-decisive rounds beginning with round r. There are two cases.

Case 1: $(c < f')$ Let p be an arbitrary correct processor. Round $r+c$ is decisive because $r+c \le t$. It follows from the definition of decisive round that there is a value v and a set S such that S contains

$\lfloor 2t+\sqrt{t}+1\rfloor$ or more correct $(r+c+1)$-dominant processors and all processors in S have FAVOR $= v$ at the end of step 12 of round $r+c$. Processor p has a tentative $(r+c+1)$-vote for v from each processor in S. Thus it has at least $\lfloor 2t+\sqrt{t}+1\rfloor$ tentative $(r+c+1)$-votes for v and at most $2t$ tentative $(r+c+1)$-votes for any $v' \neq v$. By Lemma 9 it has at least $2t+1$ final $(r+c+1)$-votes for v and at most $2t$ final $(r+c+1)$-votes for any $v' \neq v$. It follows that processor p sets FAVOR to v in step 12 of round $r+c+1$ and ends round $r+c+1 \leq f+1$ with FAVOR $= v$.

Case 2: $(c \geq f')$ Let q be an arbitrary correct processor. By Lemma 8, $|F_{q,r+f'}| \geq f'$. By Lemma 6, $F_{q,r+f'}$ contains only faulty processors from group G. Group G contains exactly f' faulty processors. Thus the set $F_{q,r+f'}$ contains all of the faulty processors in group G and nothing else. Because q is an arbitrarily chosen correct processor, $F_{p,r+f'} = F_{q,r+f'}$ for all correct processors p. In round $r+f'$ each correct processor receives precisely the same multiset of final $(r+f')$-votes from those $(r+f')$-dominant processors not in $F_{q,r+f'}$ (because they are all correct) and "ignores" those votes from processors in $F_{q,r+f'}$. In step 12 of round $r+f'$ each correct processor applies the same decision rule to precisely the same multiset of values. Thus there is a value v such that each correct processors ends round $r+f' \leq f+1$ with FAVOR $= v$. □

We now use Lemmas 9, 10, and 11 to establish the correctness of Protocol 2.

Theorem 12: *Protocol 2 solves the agreement problem.*

Proof: We show that the agreement, validity, and termination conditions are satisfied.

Agreement condition: Step 13 of the code ensures that all correct processors decide by round $t+1$. Let r be the first round in which a correct processors decides. There are two cases. The first case is $r \leq t$. Suppose that processor p decides v in round r. Step 13 ensures that processor p has at least $\lfloor 3t+\sqrt{t}+1\rfloor$ tentative r-votes for v. At least $\lfloor 2t+\sqrt{t}+1\rfloor$ of these tentative r-votes are from correct processors. Thus all correct processors have at least $\lfloor 2t+\sqrt{t}+1\rfloor$ tentative r-votes for v and at most $2t$ tentative r-votes for any $v' \neq v$. By Lemma 9 all correct processors have at least $2t+1$ final r-votes for v and at most $2t$ final r-votes for any $v' \neq v$. It follows that all correct processors end round r with FAVOR $= v$. By the choice of r, all correct processors are undecided at the start of round r. By Lemma 10 all correct processors decide v in round $r+1$. The second case is $r = t+1$. By Lemma 11 there is some round $d \leq t+1$ and some value v such that all correct processors end round d with FAVOR $= v$. By the choice of r, all correct processors are undecided at the start of round r. Because $d \leq r$ all correct processors are undecided at the start of round d. By Lemma 10 all correct processors decide v.

Validity condition: Let v be the common input of all of the correct processors. All correct processors begin round 1 with FAVOR $= v$. Each correct processor has $\lfloor 3t+\sqrt{t}+1\rfloor$ or more tentative 1-votes for v and at most t other tentative 1-votes. By Lemma 9 each correct processor has $3t+1$ or more final 1-votes for v and at most t other final 1-votes. Thus each correct processor sets FAVOR to v in step 12 and decides v in step 13 of round 1.

Termination condition: Step 13 of the code ensures that all correct processors decide by round $t+1$. □

Theorem 13: *If there are $f \leq t$ faulty processors then Protocol 2 uses the following resources: $\min(t+1, f+2)$ rounds, $\lceil\sqrt{t}\rceil \cdot \lfloor 4t+\sqrt{t}+1\rfloor$ processors, $O(t^{4.5} \cdot \log|V|)$ message bits, and $O(t^{4.5} \cdot \log|V|)$ local computation steps.*

Proof: We begin with the bound on rounds. By Lemma 11 there is a round $d \leq f+1$ and a value v such that all correct processors end round d with FAVOR $= v$. By Lemma 10 all correct processors decide by round $\min(t+1, f+2)$.

The bound on processors is immediate from the statement of the protocol. In each of $t+1$ rounds a group of $O(t)$ correct processors sends messages of size $O(t \cdot \log|V|)$ to each of the $O(t^{1.5})$ processors. Thus a total of $O(t^{4.5} \cdot \log|V|)$ message bits are sent. The bound on local computation follows because a correct processor in Protocol 2 does a constant amount of work for each message bit. □

As with Protocol 1, it is possible to improve Protocol 2 by having each correct processor stop sending two rounds after it decides and it is possible to modify Protocol 2 to reduce the number of processors by a factor of two. We omit the details of both of these optimizations.

5. RECENT DEVELOPMENTS

An exciting new protocol of Moses and Waarts [11] uses $t+1$ rounds, $6t+1$ processors, and t^8 message bits. Their protocol uses fewer processors, but more message bits, than our protocol.

ACKNOWLEDGEMENT

Ajei Gopal and Sam Toueg made helpful comments on an earlier version of this paper.

REFERENCES

[1] A. Bar-Noy, "Algorithms in an Unreliable Distributed System," Ph. D. Thesis, The Hebrew University, June 1987, (in Hebrew).

[2] A. Bar-Noy, D. Dolev, C. Dwork, and H. R. Strong, "Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement," *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing,* pp. 42–51, August 1987.

[3] B. A. Coan, "A Communication-Efficient Canonical Form for Fault-Tolerant Distributed Protocols," *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing,* pp. 63–72, August 1986. (Revised as Chapter 2 of *Achieving Consensus in Fault-Tolerant Distributed Computer Systems: Protocols, Lower Bounds, and Simulations,* Ph. D. Thesis, Massachusetts Institute of Technology, April 1987.)

[4] D. Dolev, M. J. Fischer, R. J. Fowler, N. A. Lynch, and H. R. Strong, "An Efficient Algorithm for Byzantine Agreement without Authentication," *Information and Control,* Vol. 52, Nos. 1–3, pp. 257–274, January–March 1982.

[5] D. Dolev, R. Reischuk, and H. R. Strong, "Eventual is Earlier than Immediate," *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science,* pp. 196–203, November 1982. (Revised as "Early Stopping in Byzantine Agreement," IBM Research Report RJ5406–55357, 1986.)

[6] D. Dolev and H. R. Strong, "Polynomial Algorithms for Multiple Processor Agreement," *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing,* pp. 401–407, May 1982.

[7] M. J. Fischer and N. A. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency," *Information Processing Letters,* Vol. 14, No. 4, pp. 183–186, June 1982.

[8] M. J. Fischer, N. A. Lynch, and M. Merritt, "Easy Impossibility Proofs for Distributed Consensus Problems," *Distributed Computing,* Vol. 1, No. 1, pp. 26–39, January 1986.

[9] N. A. Lynch, M. J. Fischer, and R. J. Fowler, "A Simple and Efficient Byzantine Generals Algorithm," *Proceedings of the Second IEEE Symposium on Reliability in Distributed Software and Database Systems,* pp. 46–52, July 1982.

[10] L. Lamport, R. E. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems,* Vol. 4, No. 3, pp. 382–401, July 1982.

[11] Y. Moses and O. Waarts, "Coordinated Traversal: $(t+1)$-Round Byzantine Agreement in Polynomial Time," *Proceedings of the 29th Symposium on Foundations of Computer Science,* October 1988, to appear.

[12] M. Pease, R. E. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *Journal of the ACM,* Vol. 27, No. 2, pp. 228–234, April 1980.

[13] R. Reischuk, "A New Solution for the Byzantine Generals Problem," *Information and Control,* Vol. 64, Nos. 1–3, pp. 23–42, January–March 1985.

[14] T. K. Srikanth and S. Toueg, "Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms," *Distributed Computing,* Vol. 2, No. 2, pp. 80–94, 1987.

[15] S. Toueg, K. J. Perry, and T. K. Srikanth, "Fast Distributed Agreement," *SIAM Journal on Computing,* Vol. 16, No. 3, pp. 445–458, June 1987.

ON EFFICIENTLY COMPUTING FUNCTIONS OF DISTRIBUTED INFORMATION

T. V. LAKSHMAN
VICTOR K. WEI

Bell Communications Research
331, Newman Springs Road
Red Bank, NJ 07701.

ABSTRACT

Efficient algorithms are presented for collecting distributed information while performing certain operations on the information being collected. Such algorithms are useful for many distributed computing applications. Typical examples are collecting status information regarding distributed resources, decentralized task scheduling, and maintaining transaction atomicity in distributed databases. Our algorithms synthesize the required information by executing identical programs at all nodes. The programs do not use node identifiers in their local computation and do not use broadcasting of information to all nodes. The efficiency objective is to minimize the communication-delay product and to obtain tradeoffs between the two.

The amount of communication and the delay are determined by the degree and diameter of the graph determining inter-node communication. Hence small degrees and small diameters are desirable for this graph. Choice of this graph is constrained by the type of operation being performed on the information being collected. When operations performed on the information are idempotent (MAX, MIN, etc), folded generalized n-gons and de Bruijn graphs achieve close to or exactly the lower bound of $\Omega(N \log^2 N)$ on the communication-delay product. For performing non-idempotent operations on the collected information, difficulties arise due to multiple paths between nodes in the communication graph. Information arriving at a node along a path may already have arrived previously along a shorter path and it is difficult to distinguish new information without using node identifiers to determine the origin of information. A prime result in the paper is the identification of metrically regular graphs (which include the widely used hypercube interconnections) as graphs useful in developing efficient algorithms for performing sum-like operations (Abelian group operations, computing elementary symmetric functions). To our knowledge, our algorithms using metrically regular graphs are the only algorithms for performing sum-like operations which do not use node identifiers. Our best algorithm has a communication-delay product of $O(N \log^3 N)$, and obtaining an optimal $\Omega(N \log^2 N)$ algorithm remains an open problem. Other related open problems are also identified.

1. INTRODUCTION

We consider the paradigmatic distributed computation requirement of gathering some information from all nodes in the system, operating on (computing a function of) the gathered information, and making the result known to all nodes. The information, for example, can be the number of unfinished tasks at each node with the function computed being the sum. Similar computations are needed for coordination of decentralized checkpoints [13], and in decentralized commit protocols [12]. The efficiency objective is to minimize the communication-delay product and to obtain tradeoffs between the two.

A straightforward solution is to send all the information to one node, compute the required function, and distribute the results to all nodes. However, this solution is neither robust nor symmetric and in some situations no node may be willing to be the central controller - for example, if the nodes are personal workstations. A straightforward symmetric solution is for all nodes to send their information to all other nodes and perform function computation locally. However, for N nodes the communication- delay product is $O(N^2)$ while the lower bound on the communication-delay product for symmetric protocols is $\Omega(N \log^2 N)$.

For a wide range of functions, we present efficient symmetric protocols minimizing the communication-delay product or demonstrating tradeoffs between the two. All nodes execute identical programs and the programs do not use node identifiers in their computations. As in [2], instead of broadcasting, nodes collect the required information by using rounds of message exchanges. For our protocols, in each round a function of information received in the previous round (or the initial information) is computed and forwarded to other nodes. Unlike in [2], in our protocol nodes do not communicate with different nodes in different rounds. Instead, each node communicates with a fixed set of other nodes. A node only needs to know this set and does not have to distinguish amongst the members of this set. We take the number of messages used in executing the protocol (message-complexity) as the measure of the amount of communication. Local computation is assumed to be small ($O(1)$) and the delay is taken to be the number of rounds of message exchanges.

We specify the inter-node communication pattern by a graph (called the communication graph). Proper choice of this graph is crucial to both the correctness and efficiency of the protocol. Regular graphs (to satisfy symmetry) of small degree and diameter are desirable since the degree and diameter determine the message-complexity delay product. By varying the degree and diameter of the communication graph we can obtain tradeoffs between communication and delay, but the constraint of not using node identifiers (which could be used to tag the origin of information) puts constraints on both the functions that can be evaluated and the graphs that can be used.

There can be several paths[1] in the communication graph by which the information at a node reaches another node. For simple functions such as MAX, MIN, AND, OR, etc., the communication graph can be any regular graph because the effect of information from one node being included several times in the computation of other nodes is not important. Optimal protocols achieving close to or exactly the lower bound on the communication-delay product are obtained using generalized n-gons and de Bruijn graphs (graphs of bounded degree and $O(N \log N)$ diameter). Commit protocols [12] are important special cases.

For computing sum-like functions, the effect of multiple paths cannot be ignored and hence arbitrary regular graphs cannot be used as communication graphs. An important contribution of this paper is the identification of a novel graph property - metric regularity - useful for sum-like function computation. Our protocol correctly computes sum-like functions for any communication graph which is metrically regular. The widely used hypercubes are metrically regular and are important special cases. Other metrically regular graphs are graphs constructed from finite projective planes, generalized n-gons, etc. Our best protocol has a message-complexity delay product of $O(N \log^3 N)$. Obtaining an optimal $O(N \log^2 N)$ protocol remains an open problem. Another open problem is whether metrically regular graphs are necessary for these protocols.

The next section describes the protocols. For clarity of exposition specific functions are used rather than a canonical protocol even though all function specific protocols are adaptations of a canonical protocol. For different functions, the node programs and communication graph characteristics are specified along with correctness proofs. Section 3 outlines construction of communication graphs for various communication and delay parameters. Section 4 has concluding remarks and directions for future work.

2. PROTOCOL DESCRIPTION

In describing the protocols, the N computing nodes are referred to by the lower case letters $i, j, k, \ldots$, etc. The initial information at node i is denoted by V_i and is referred to as the value of node i. The communication graph is denoted by $G(V, E)$, where V the vertices of the graph are the computing nodes and an edge $(i, j) \in E$ iff nodes i and j exchange messages with each other in the protocol. From the protocol requirements, the communication graph is always a regular graph. The node degree is denoted by R and the diameter by D.

The algorithm proceeds in rounds of message exchanges. In the first round, every node sends its initial value to all its neighbors. In each subsequent round, a node waits to receive messages from all its neighbors. Upon receiving all messages, it computes a function of the received values and any previous values that it might have stored. It then sends this newly computed value to all its neighbors. The number of rounds is a multiple of the diameter of the graph. In the last round, a node instead of sending the newly computed value to all its neighbors goes into a final state where it computes the required result from all the previously stored values. In our delay analysis we have assumed that a node can exchange messages with all of its neighbors in $O(1)$ time. For communication graphs of bounded degree this assumption is perfectly valid. For other graphs, if this assumption is relaxed then the delay is multiplied by the node degree R.

---

[1] We define a *path* of a graph to be an alternating sequence of vertices and edges, beginning and ending in vertices, in which each edge is incident with the two vertices immediately preceding and following it. Note that all the vertices need not be distinct.

Idempotent functions: Repeated application of an idempotent function (MAX, MIN, AND, OR, etc) to the same argument does not change the result of the function evaluation making multiple inclusions of a node's value in another node's computation irrelevant. Multiple paths between nodes in the communication graph can therefore be ignored. The choice of communication graphs is then dictated only by message-complexity and delay requirements. We illustrate idempotent function computation by taking the information at each node to be integers and computing their maximum. The algorithm for each node is as follows:

Protocol for MAX

```
begin
 Value := Initial_Node_Value
 for Round := 1 to D do
  begin
   Send Value to all neighbors in communication graph
   Receive messages from all neighbors
   Value := MAX(Value, values received in messages from neighbors)
  end
end.
```

At the end of the first round, each node will have the maximum of the initial values of itself and all its neighbors. At the end of the second round, each node will have the maximum of the initial values of itself, all its neighbors, and all nodes at distance 2. In general, at the end of round t each node will have the maximum of values of all nodes at distance t or less from it. (This can be easily proved by induction on rounds. The proof is omitted for brevity.) Consequently, D rounds are sufficient for the protocol to correctly compute the maximum of the initial values.

Other idempotent functions can be computed by similar protocols. Other functions such as the kth largest number can also be computed. For example, the second largest number can be computed in $2D$ rounds. MAX is computed in the first D rounds. In the next D rounds, MAX of the remaining values is computed. Protocols to compute kth largest number more efficiently remain an open problem. An important practical example of computing idempotent functions is commit protocols.

Sum-like functions: The MAX protocol cannot be directly applied to computing the sum (i.e. we cannot just compute the sum of the received values instead of the maximum) because the initial value of one node may get added several times at another node due to multiple paths in the communication graph between the two nodes. The protocol has to compensate for these multiple inclusions of a node's value in another nodes computation. This can be easily done if node identifiers indicating the origin of information are used to filter out multiple occurrences of the same information. However, use of node identifiers is forbidden by the protocol requirements. Our protocol relies only on the graph properties of the inter-node communication pattern to compute the sum correctly despite duplicate information. The protocol correctly computes the sum whenever the communication graph is metrically regular[2]. For $D = 2$, we also show that our protocol correctly computes the sum only if the communication graph is metrically regular. The question of whether metrically regular graphs are necessary for sum computation protocols when $D > 2$ remains unanswered. An important family of metrically regular graphs are the binary hypercubes used as the interconnection network in many current parallel computers [6, 11]. Other metrically regular graphs are graphs constructed from finite projective planes and generalized polygons. Our best protocol has a message-complexity delay product of $O(N\log^3 N)$. Achieving the lower bound of $\Omega(N\log^2 N)$ remains open. The difficulty is in constructing a metrically regular graph of bounded degree and $O(logN)$ diameter. The protocol for computing sums is given below.

---

[2] A metrically regular graph is a graph in which for any two vertices x and y with distance k between them, the number of vertices z at distance i from x and distance j from y is a constant $c(i,j,k)$ independent of the choice of x and y

Protocol for SUM

```
begin
 X[0] := Initial_Node_Value (* X array stores intermediate values *)
 for Round := 1 to D do
  begin
   Send X[Round - 1] to all neighbors in communication graph
   Receive messages from all neighbors
   X[Round] := sum of values received in messages from neighbors
  end
 Sum := $\sum_{t=0}^{D} \alpha_t X(t)$
end.
```

The coefficients of the linear combination, α_ts, used in the final sum computing step are identical for all nodes (otherwise all nodes will not be executing identical programs). The existence of such a linear combination independent of the vertices is guaranteed by the properties of metrically regular graphs and is shown later. The α_ts depend on the communication graph and are known a priori by the nodes.

Let $n(i,j,d,t)$ denote the number of paths (in the communication graph) of length t between nodes i and j with distance between i and j equal to d. The relation needed to compute the α_ts from parameters of the communication graph is:

$$X_i(t) = \sum_{d=0}^{t} \sum_{\substack{\text{all } j \text{ at} \\ \text{distance } d}} n(i,j,d,t) * V_j, \text{ where } 1 \le t \le D.$$

The relation can be formally proved by induction on t. However, the validity can be easily seen by imagining an equivalent protocol. Suppose there is no constraint on message lengths. Each node upon receiving messages from all its neighbors computes the sum of the received values. But instead of sending partial sums, it sends a message created by by appending all messages received in the previous round. A value V_j traces a path of length equal to the round number, and the number of times V_j arrives at node i in round t is equal to the number of paths of length t between i and j. V_j is therefore included $n(i,j,d,t)$ times in $X_i(t)$.

Since all nodes must execute identical programs, the α_ts should be the same for all nodes. This is the case if $n(i,j,d,t) = n(d,t)$, i.e. the number of paths of length t between any two nodes at distance d does not depend on the pair of nodes involved but depends only on d and t. Then we have

$$X(t) = \sum_{d=0}^{t} n(d,t) \sum_{\substack{\text{all } j \text{ at} \\ \text{distance } d}} V_j = \sum_{d=0}^{t} n(d,t) * S(d).$$

$S(d)$ denotes the sum of initial values of all nodes at distance d from the given computing node. The above relation expresses the Xs as a linear combination of Ss (sums of nodes at different distances). Conversely the Ss can expressed as linear combinations of the Xs. In matrix form, this can be written as

$\mathbf{S} = \mathbf{n}^{-1}\mathbf{X}$, (where $\mathbf{S} = [S(1), S(2), .. S(k)]$, $\mathbf{n} = n(t,d), 0 \le d, t \le D$, and $\mathbf{X} = [X(1), X(2), .. X(k)]$) because $\mathbf{n}$ is lower triangular with non-zero diagonal elements. (The matrix is lower triangular because between two nodes there cannot be any path shorter than the distance between them. The diagonal elements are non-zero because if two nodes are at distance d then there has to be at least one path of length d between them.) The α_ts in the protocols are the coefficients in the linear combination and, as seen above, depend on the number of paths of different lengths in the communication graph. For example, with a three-dimensional hypercube, we have

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3 & 0 & 2 & 0 \\ 0 & 5 & 0 & 6 \end{bmatrix} \begin{bmatrix} S(0) \\ S(1) \\ S(2) \\ S(3) \end{bmatrix}$$

Inverting the n matrix, we can express the sums as

$$\begin{bmatrix} S(0) \\ S(1) \\ S(2) \\ S(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/2 & 0 & 1/2 & 0 \\ 0 & -5/6 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix}$$

The next theorem shows metrically regular graphs have the property on paths that we need. $c(x,y,i,j,k)$ denotes the number of nodes z such that $d(x,z)$ (the distance between x and z) equals i and $d(z,y)=j$ and $d(x,y)=k$.

Theorem 1.

Given a graph $G(V,E)$, we have

(1) If $n(x,y,d,t)=n(d,t)$ for all $0 \le d \le t \le D$, then $c(x,y,i,j,k)=c(i,j,k)$ for all $i+j \le D$.

(2) If $G(V,E)$ is metrically regular, then $n(x,y,d,t)=n(d,t)$ for all $0 \le d \le t$

Proof: Given in the appendix.

□

The above theorem shows that metrically regular communication graphs are sufficient for our protocols to compute sums. Theorem 3 shows that for $D=2$, sums can be computed only if the communication graphs are metrically regular.

Theorem 2.

Metrically regular graphs are necessary to compute sums when $D=2$.

Proof: Given in the appendix.

□

Abelian groups and symmetric functions: The sum computation protocol can be modified to compute several other sum-like functions. Products can be computed by using the sum protocol to compute the sum of the logarithms of the initial values. The product can then be obtained from this sum. In fact, the sum protocol can be used to perform any Abelian group operation. This is because every finite Abelian group can be decomposed into a direct sum of primary cyclic groups [10].

Next, we show that the sum protocol can be used to compute symmetric functions. A function or polynomial on $V_1, V_2 \cdots V_N$ is symmetric if and only if the function's value is unchanged by any permutation of the arguments $V_1, V_2 \cdots V_N$. The elementary symmetric functions $F_1, F_2 \cdots F_N$ are defined as the following polynomials:

$$F_1 = V_1 + V_2 + \cdots + V_N$$
$$F_2 = V_1V_2 + V_1V_3 + \cdots$$
$$F_3 = V_1V_2V_3 + V_1V_2V_4 + \cdots$$
$$F_N = V_1V_2V_3 \cdots V_N$$

where F_k is the sum of all $\frac{N!}{(N-k)!k!}$ products combining k distinct V_is. Let

$$f_0 = N$$
$$f_1 = \sum_{i=1}^{N} V_i$$
$$f_2 = \sum_{i=1}^{N} V_i^2 \quad \cdots\cdots\cdots$$
$$f_k = \sum_{i=1}^{N} V_i^k.$$

The f functions can be computed directly using the sum protocol. To compute f_k, each node uses the sum protocol but sends V_i^k instead of V_i. Note that a complete execution of the sum protocol is used for computation of each f_k. The elementary symmetric functions can be computed locally at each node from the f_ks using Newton's formula given below:

$$(-1)^k kF_k + (-1)^{k-1} f_1 F_{k-1} + (-1)^{k-1} f_2 F_{k-2} + \ldots\ldots = 0, \ k = 1,2,\cdots$$

F_k is defined to be zero for $k > N$ and $k < 0, F_0 = 1$

It follows that the sum protocol can be used to evaluate every symmetric polynomial, since all such polynomials can be expressed as a polynomial in a finite number of symmetric functions. [9]. However to compute F_k, the sum protocol must be executed k times to compute $f_1 f_2 \cdots f_k$. More efficient computation of elementary symmetric functions remains an open problem.

3. COMMUNICATION AND DELAY

In each round, the protocol requires every node to send messages to all its neighbors in the communication graph. The total number of messages used is therefore $N * R * D$. A round is assumed to complete in time independent of the number of nodes. Hence, the delay is $O(D)$. The message-complexity delay product is $O(N * R * D^2)$. For any D, the lower bound on the message-complexity is given by the Moore bound [7] which states that $N \le 1 + R + \cdots + R(R-1)^{D-1}$. This implies that R is $\Omega(N^{1/D})$ and the lower bound on the number of messages is $\Omega(N * N^{1/D} * D)$. This lower bound is in fact independent of the protocol because in any protocol the value at a node must causally affect the value at any other node. Let R be the maximum number of nodes with which a node i communicates. At the end of k rounds, V_i can affect the computation of at most $1 + R + \cdots + R(R-1)^{k-1}$ nodes. To terminate in D rounds, we need $N \le 1 + R + \cdots + R(R-1)^{D-1}$ implying that R is $\Omega(N^{1/D})$. For symmetric protocols, R is the same for all nodes and the message-complexity is $\Omega(N * N^{1/D} * D)$.

The best protocols are obtained by coming as close to the Moore bound as possible. The Moore bound itself is known to be unachievable except in the following cases.

(1) $D = 1$: N-cliques
(2) $R = 2$: $(2D+1)$-cycles
(3) $D = 2, R = 3$: Peterson graph
(4) $D = 2, R = 7$: Hoffman-Singleton graph
(5) possibly $D = 2, R = 57$: no known construction yet.

Refer [4] for more information on Moore graphs. Table 5 summarizes the parameters relevant to communication and delay (N, R, D) for various graphs suitable to our protocols. (q is a prime power. $q \ne 2$ for the last folded 4-gon.)

Graph	N	R	D
Cliques	$R(R+1)$	R	1
Folded Projective Plane	$(q^2+q+1)^2$	$(q+1)^2$	2
Folded 4-gon	$(q+1)^2(q^2+1)^2$	$(q+1)^2$	3
Folded 4-gon	$(q+1)(q^2+1)(q^3+1)^2$	$(q+1)(q^2+1)$	3
Folded 4-gon	$(q^2+1)(q^3+1)(q^5+1)^2$	$(q^2+1)(q^3+1)$	3
Folded 4-gon	$q^5(q+2)\, q(q+2)$		3
Folded 6-gon	$(q+1)^2(q^4+q^2+1)^2$	$(q+1)^2$	5
Folded 6-gon	$(q^3+1)^2(q^8+q^4+1)^2$	$(q+1)(q^3+1)$	5
Folded 8-gon	$2^{15m}+(2^{12m}-1)(1+\frac{2^m+1}{2^{4m}-1})$	$(2^m+1)(2^{2m}+1)$	7
de Bruijn	s^r	$2s$	r
Hypercube	q^k	$k(q-1)$	k

Table 5. Communication Graph Parameters

Details of the graphs are given below.

Hypercubes: D-dimensional hypercubes can be used for all our protocols. A D-dimensional hypercube with side s is a graph $G(V,E)$ with $|V| = N = s^D$ vertices corresponding to all D-tuples $x_1, x_2, \cdots, x_D$, each $x_i \in \{1, 2, \cdots, s\}$. Two vertices are adjacent if and only if the corresponding pair of D-tuples differ in exactly one coordinate. It can be verified that the hypercube is a metrically regular graph with diameter D and degree $D(s-1) = O(DN^{1/D})$, and hence achieving a communication complexity of $O(D^2 N^{1+1/D})$. Since communication complexity is parametrized by delay, we get a family of protocols in which communication can be traded off for delay. The minimum communication-delay product for hypercubes is $O(N \log^3 N)$ and is obtained by setting $D = \log N$.

De Bruijn Graphs: De Bruijn graphs give asymptotically optimal protocols for computing functions which do not require metric regularity. The de Bruijn graph [3] $G = (Q^D, E)$ is a graph with vertex set $Q = \{0,1, \cdots ,q-1\}$ and edge-set E where an edge $(\alpha,\beta) \in E$ if and only if $\alpha = a\gamma$ and $\beta = \gamma b$ for some $a, b \in Q$, $\gamma \in Q^{m-1}$, $q \geq 0, D \geq 2$. (Q^D denotes the cross-product of the set Q with itself D times.) This graph has q^D vertices, maximum degree $2q$, and diameter D. However, it is not a regular graph.

From the construction of G we see that for each vertex $(a, a, \cdots, a)$, $a \in Q$, one of the edges from this vertex is a self-loop, and for each pair of vertices of the form $(a, b, a, b, \cdots)$, $(b, a, b, a, \cdots)$ there is a double-edge between the two. Therefore, each vertex $(a, a, \cdots, a)$ has two self-loops - one corresponding to the first case above, and the second corresponding to the second case with $a = b$. Removal of these self-loops results in q vertices with degree only $2q-2$. Removal of the other double edges (corresponding to $a \neq b$) results in $q * (q-1)$ vertices with degree $2q-1$. To obtain a regular graph from this graph, add q vertices x_i, $i = 0,1, \cdots ,q-1$ with q edges between x_i and vertices $(\alpha, i, \alpha, i, \cdots)$ for any $\alpha \in Q$. The x_i have degree q, and there are q vertices (previously of degree $2q-2$) of degree $2q-1$. All other vertices have degree $2q$. Add a vertex x_q connected to all vertices whose degrees are not $2q$. The degree of x_q is $2q$, the degree of each x_i, $i = 0,1, \cdots ,q-1$ is $q+1$, and all other vertices have degree $2q$. Add an edge between every x_i and x_j, $i \neq j, i, j = 0,1, \cdots ,q-1$. Now, all vertices have degree $2q$. The graph has $N = q^D + q + 1$ vertices, degree $2q = constant$, and diameter $D = O(logN)$ For idempotent functions, de Bruijn graphs give optimal protocols with $O(logN)$ delay and $O(N \log N)$ message complexity.

Projective Planes: For $D = 2$, finite projective planes [1] and folded finite projective planes are alternatives to hypercubes. However, folded finite projective planes are not metrically regular and so cannot be used for sum-like function computation. Projective planes can be constructed only for certain N values. Protocols using projective planes have a message-complexity delay product of $O(N\sqrt{N})$. When they exist, the constants for folded finite projective planes are lower than those for 2-dimensional hypercubes. For example, when $N = 49$ the message-complexity for protocols using folded finite projective planes is $49 * 9 * 2 = 882$ while it is $49 * 14 * 2 = 1372$ for the 2-dimensional hypercube. (A node is assumed to send messages to itself.) Finite projective planes are a special case of generalized n-gons.

Generalized n-gons and Folded n-gons: A generalized n-gon is a bipartite graph with diameter n and girth[3] $2n$. Generalized n-gons are derived from finite geometries. The nodes in one set of the bipartite graph are called *points* and those in the other set are called *lines*. When all nodes have degree 2, the bipartite graph represents the incidence relationships in a standard n-gon.

If all node degrees are at least three, the graph is called a *thick generalized n-gon*. In this case, one can prove that the graph is an $(s+1, t+1)$-regular bipartite, i.e. every line is incident to $s+1$ points and every point is incident to $t+1$ lines. This n-gon is called an (s, t) n-gon. The generalized 2-gons are the complete bipartite graphs. The generalized 3-gons (triangles) are non-degenerate finite projective planes. Feit and Higman [5] proved that thick generalized n-gons exist only for $n = 2,3,4,6$, and 8. The parameters of several thick generalized n-gons was given previously. Their constructions are developed from finite geometries in very complicated ways. For details, see Kantor [8].

The generalized n-gons are metrically regular when $s = t$ and hence can be used for sum-like function computation. The constants in the message-complexity can be reduced by using folded generalized n-gons. However, folding loses the metric regularity property thereby restricting applicability. Let $G(V, E)$ be a thick generalized n-gon whose nodes are partitioned into points and lines. The folded n-gon $G'(V', E')$ consists of nodes $\{(p, l): p$ is a point and l is a line in $G\}$ and edges $\{[(p_1, l_1), (p_2, l_2)]$: if p_1 is adjacent to l_2 and p_2 to l_1 in $G\}$. Note that $|V'| = |V|^2$, $diameter(G') = diameter(G) - 1$, and G' is $(s+1)(t+1)$-regular if G is $(s+1)(t+1)$-regular bipartite.

---

[3] A bipartite graph is one whose nodes can be partitioned into two sets with no adjacent pairs in the same set. The girth of a graph is the length of a smallest circuit.

Parameters for folded n-gons were listed in Table 5.

CONCLUDING REMARKS

We have developed efficient protocols for collecting distributed information while performing certain operations on the information being collected. Our protocols for idempotent operations (and its applications such as commit protocols) improve previous protocols and this is achieved by using sparser communication graphs such as those constructed from folded generalized n-gons. Sum-like operations such as elementary symmetric functions, symmetric polynomials and Abelian group operations are intrinsically harder to compute in a decentralized manner without using node identifiers in the computation. To our knowledge, our protocols for sum-like function computation are the only known protocols which do not use node identifiers. We use a novel graph property - metric regularity - to obtain these protocols. The communication delay product is within a $\log N$ factor of the lower bound. Achieving the lower bound of $\Omega(N\log^2 N)$ remains an open problem. The difficulty is in constructing metrically regular graphs of bounded degree and $O(\log N)$ diameter. Another open problem is whether metrically regular graphs are necessary for our protocols. We prove necessity only for the case $D = 2$. For $D = \log N$ if metrically regular graphs are not necessary, then the lower bound might be differently achieved. Several other open problems remain. Our protocol for symmetric polynomials of degree k requires $O(kN\log^2 N)$ messages. More efficient computation of these polynomials (and consequently elementary symmetric functions) is not known at present. Our protocols can be used to compute functions such as k^{th} largest number. This uses $O(kN\log N)$ messages and for computing medians this is less efficient than the straightforward $O(N^2)$ protocols. Efficient computation of these functions also remains open.

APPENDIX

Here we prove theorems 1 and 2.

Theorem 1.
Given a graph $G(V,E)$, we have

(1) If $n(x,y,d,t) = n(d,t)$ for all $0 \le d \le t \le D$, then $c(x,y,i,j,k) = c(i,j,k)$ for all $i + j \le D$.

(2) If $G(V,E)$ is metrically regular, then $n(x,y,d,t) = n(d,t)$ for all $0 \le d \le t$

Proof:
Assume G is metrically regular. Then for $t = 1$,

$$n(x,y,d,1) = \begin{cases} 1 & \text{if } d = 1 \\ 0 & \textit{otherwise (including } d < 1\textit{)} \end{cases}$$

The proof is by induction on t. Assume $n(x,y,d,t') = n(d,t)$ whenever $t' < t$. Then

$$n(x,y,d,t) = \sum_{\substack{z:d(x,z)=1 \\ d(z,y)=d-1}} n(z,y,d-1,t-1) + \sum_{\substack{z:d(x,z)=1 \\ d(x,z)=d}} n(z,y,d,t-1)$$

$$+ \sum_{\substack{z:d(x,z)=1 \\ d(z,y)=d+1}} n(z,y,d+1,t-1)$$

$$\overset{(1)}{=} |\{z:d(x,z)=1, d(z,y)=d-1\}|\, n(d-1,t-1)$$

$$+ |\{z:d(x,z)=1, d(z,y)=d\}|\, n(d,t-1)$$

$$+ |\{z:d(x,z)=1, d(z,y)=d+1\}|\, n(d+1,t-1)$$

$$\overset{(2)}{=} c(1,d-1,d)n(d-1,t-1) + c(1,d,d)n(d,t-1) + c(1,d+1,d)n(d+1,t-1)$$

$= n(d,t)$, independent of choice of x and y.

This establishes one direction of the theorem. Equality (1) comes from the induction hypothesis and (2) from metric regularity.

To prove the other direction, assume $n(x,y,d,t) = n(d,t)$ for all $0 < d < t < D$. Since the parameters i, j, and k denote distances between nodes they must satisfy the triangular inequality. Therefore, $c(x,y,i,j,k) = c(i,j,k) = 0$ if $i < 0, j < 0, k < 0, i + j < k, i + k < j$, or $j + k < i$.

We proceed by induction on the integer $i + j - k$. Let i,j,k be given and $i + j = k$. Let x,y be nodes with $d(x,y) = k$. By considering nodes at distance i from x, the number of paths between x and y of length k can be expressed as

$$\begin{aligned} n(x,y,k,k) &= \sum_{\substack{z:d(x,z)=i \\ d(z,y)=j}} n(x,z,i,i)n(z,y,j,j) \\ &= c(x,y,i,j,k)n(i,i)n(j,j). \end{aligned}$$

Therefore, $c(x,y,i,j,k) = c(i,j,k) = \dfrac{n(k,k)}{n(i,i)n(j,j)}$, independent of x and y when $i + j - k = 0$.

Now assume that $c(x,y,i',j',k') = c(i',j',k')$ whenever $i' + j' - k' < i + j - k$. Let x and y be two nodes with $d(x,y) = k$. Let $S = \{(i',j') : i' \le i, j' \le j$, and i',j',k satisfy the triangular inequality $\}$. By considering nodes at distance i from x (note that $i,j,k \le D$), we can write the number of paths from x to y of length k as

$$\begin{aligned} n(x,y,k,i+j) &= \sum_{(i',j') \in S} \sum_{\substack{z:d(x,z)=i' \\ d(z,y)=j'}} n(x,z,i',i)n(z,y,j',j) \\ &= \sum_{(i',j') \in S} c(x,y,i',j',k)n(i',i)n(j',j) \\ &= c(x,y,i,j,k)n(i,i)n(j,j) + \sum_{\substack{(i',j') \in S \\ (i',j') \ne (i,j)}} c(i',j',k)n(i',i)n(j',j) \end{aligned}$$

Note that for every $(i',j') \in S$, and $(i',j') \ne (i,j)$, we have $i' + j' - k' < i + j - k$. This yields the last equality by the induction hypothesis. Since all other terms in the equality are independent of x and y, we have $c(x,y,i,j,k) = c(i,j,k)$.

$\square$

Theorem 2.
Metrically regular graphs are necessary to compute sums when $D = 2$.
Proof:
Consider a node x. The values computed and stored at node x at the end of rounds 0, 1, and 2 (the protocol terminates after round 2 since $D = 2$) are denoted by by X_0, X_1, and X_2. The sum is computed by $S = \alpha_0 X_0 + \alpha_1 X_1 + \alpha_2 X_2$. Note that
$X_0 = V_0$, $\quad X_1 = \sum_{d(x,y)=1} V^y{}_0$, and

$$X_2 = V_0\, degree(x) + \sum_{d(x,y)=1} V^y{}_0 n(x,y,1,2) + \sum_{d(x,y)=2} V^y{}_0 n(x,y,2,2)$$

where $V^y{}_0$ is the initial value of node y.

The coefficient of V_0 in S is $\alpha_0 + \alpha_2\, degree(x)$ and this should be equal to one. Therefore, $degree(x)$ must be constant and G is a regular graph, i.e. $n(x,y,1,1) = n(1,1)$. For y such that $d(x,y) = 2$, the coefficient of $V^y{}_0$ in S is $\alpha_2 n(x,y,2,2)$. So $n(x,y,2,2) = n(2,2)$ is a constant. For y with $d(x,y) = 1$, the coefficient of $V^y{}_0$ in S is $\alpha_1 + \alpha_2 n(x,y,1,2) = 1$. So $n(x,y,1,2) = n(1,2)$ a constant.

By Theorem 1, $C(x,y,i,j,k) = C(i,j,k)$ for all $i+j \leq 2$. It remains to show that $C(x,y,i,j,k) = C(i,j,k)$ for $i+j > 2$. Consider node x. We have
$C(x,x,2,2,0) = N - C(x,x,0,0,0) - C(x,x,1,1,0) = N - 1 - R = C(2,2,0)$.

By classifying all neighbors of x according to their distance to y, we obtain
$C(x,y,1,2,1) = R - C(1,0,1) - C(1,1,1) = C(1,2,1)$
$C(x,y,1,2,2) = R - C(1,1,2) = C(1,2,2)$

Similarly by classifying all vertices at distance 2 from x according to their distance to y, we obtain
$C(x,y,2,2,1) = C(2,2,0) - C(2,1,1) = C(2,2,1)$
$C(x,y,2,2,2) = C(2,2,0) - C(2,0,2) - C(2,1,2) = C(2,2,2)$

□

Therefore, for $D = 2$ the communication graph is metrically regular.

REFERENCES

1. A. A. Albert and R. Sandler, *An Introduction to Finite Projective Planes*, Holt, Rinehart and Winston (1968).
2. N. Alon, A. Barak, and U. Manber, "On Disseminating Information Reliably Without Broadcasting," *Proceedings of the 7th International Conference on Distributed Computing Systems*, pp. 74-81 (Sept. 1987).
3. J. C. Bermond, J. Bond, M. Paoli, and C. Peyrat, "Graphs and Interconnection Networks: Diameter and Vulnerability," in *Lecture Notes Series #82, Surveys in Combinatorics*, London Mathematics Society (1983).
4. F. R. K. Chung, "Diameters of Communication Networks," *1984 AMS Short Course on Mathematics of Information Processing*, (1984).
5. W. Feit and G. Higman, "The Non-existence of Certain Generalized Polygons," *Jounal of Algebra*, pp. 114-131 (1964).
6. M. T. Heath, "Hypercube Multiprocessors 1986," *Society for Industrial and Applied Mathematics*, (1986).
7. A. J. Hoffman and R. R. Singleton, "On Moore Graphs with Diameter 2 and 3," *IBM Journal of Research and Development* 4 pp. 497-504 (1960).
8. W. Kantor, "Generalized Polygons," *Preprint (to appear in Springer Lecture Notes)*, ().
9. G. A. Korn and T. M. Korn, "Section 1.4-3," pp. 11 in *Mathematical Handbook for Scientists and Engineers*, McGraw Hill (1968).
10. J. J. Rotman, "Theorem 4.6 (Basis Theorem)," pp. 54 in *The Theory of Groups: An Introduction*, Allyn and Bacon Inc. (1973).
11. C. L. Seitz, "The Cosmic Cube," *Communications of the ACM* **28,1** pp. 22-33 (Jan. 1985).
12. D. Skeen, "Nonblocking Commit Protocols," *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 133-142 (1981).
13. S. H. Son, "Efficient Decentralized Checkpointing in Distributed Database Systems," *Proceedings of the 21st Hawaii International Conference on System Sciences*, (Jan. 1988).

ON FINDING STRONGLY CONNECTED COMPONENTS DISTRIBUTIVELY

K. Brahmadathan and S. Venkatesan
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260

1. INTRODUCTION

Given a directed graph G=(V,E) consisting of the vertex set V and edge set E, the vertices in V can be partitioned into disjoint sets V_1, ..., V_k such that two nodes u and v are in the same set if and only if there is a (directed) path from u to v and a (directed) path from v to u. For each V_i, let $E_i \subseteq$ E be the set of edges connecting nodes within V_i. The subgraphs $C_i=(V_i,E_i)$ are called the *strongly connected components* of G. A graph is said to be *strongly connected* if it has only one strongly connected component.

Many applications in distributed system can be modeled by means of directed graphs. Typically, the nodes are identified with processes while the edges denote some relationship between the processes. For instance, in a distributed database, the nodes may represent transactions while the edges denote dependencies: facts such as a transaction is waiting on a resource held by another.

Finding the strongly connected components is an important graph-theoretic problem [1], and strongly connected components are useful in many applications in distributed systems such as deadlock detection. As shown subsequently, an efficient distributed algorithm for *knot detection* can be developed using strongly connected components. The reader is referred to [2, 6] for applications of this important problem.

In this paper, we study the problem of finding the strongly connected components of a distributed system and present a message-efficient distributed algorithm. As an application, we demonstrate how the knot detection problem may be efficiently solved.

The organization of this paper is as follows. The model of computation is presented in section 2. A message-efficient algorithm is presented in section 3. Section 4 contains an efficient knot detection algorithm. Section 5 concludes the paper.

2. MODEL OF COMPUTATION

The following model of computation is used in this paper. The distributed computer system is a network of nodes, each node having a distinct label. Each node has a set of communication links and each link connects a given node with another node of the network. These communications links are bidirectional - both of the end nodes know the existence of the link and messages may be transmitted in either direction. Call the node p a **neighbor** of the node q if they are connected by a communication link. At each node of the distributed system, there is a processor with a small amount of local memory. The local memory of one processor cannot be shared by another processor. The only way that two processors can share some information is by explicitly sending messages over the links of the network. Thus there is no global memory in the system.

The processors and the communication links are asynchronous (that is, the processors and the communication links incur arbitrary but finite delays in executing their function). Thus, there is no fixed bound on the time taken for a message to be transmitted across *any* link in the computer system. Also, each processor incurs an arbitrary but finite delay in executing a step. Processors and links are assumed to be non-faulty. In particular, links are assumed to deliver the messages sent over them without modifications, and in the order in which they are transmitted. (i.e. if a node v_1 sends two messages m_1 and m_2 to a neighbor v_2 on the same link in that order, then v_2 receives the two messages in the order m_1, m_2). We consider the point-to-point communication model where a processor can generate a single message at a time and send it to one of its neighbors.

More formally, the distributed system is represented by a graph G=(V,E) where V is the set of nodes or vertices, $E \subseteq V \times V$ is the set of directed edges such that $(u,v) \varepsilon E$ if and only if u and v are related (for example, in deadlock detection, u must have made a request to v). If there is an edge $(u,v) \varepsilon E$, a bidirectional link is assumed to exist between the two nodes. Readers familiar with [6] may observe that our model is similar to the one reported in that work. For each $(u,v) \varepsilon E$, v is a *successor* of u, and u is a *predecessor* of v. Let the cardinality of the node set be $|V|$ and the cardinality of the edge set be $|E|$. Our algorithm assumes the existence of a leader. In case a leader is not available, the minimum spanning tree algorithm of Gallager et al. [4] can be run to elect a leader using O(|E|+|V|*log*|V|) messages.

Messages are sent by using the primitive

send(*source*, *destination*, parameters)

where source is the identity of the node sending the message while destination is the identity of the node to whom the message is destined. The parameters consist of the actual data that is sent. Similarly, messages are received using the primitive

receive(*source*, *destination*, parameters)

The performance of the distributed algorithm is measured using the worst case *communication* complexity and the worst case *bit* complexity. While the *message complexity* counts the total number of messages sent over the communication links, the *bit complexity* counts the total number of *bits* sent over the links. To avoid trivial solutions, we assume that the messages are of size O(log|V|) bits.

3. STRONGLY CONNECTED COMPONENTS

In this section, we present a distributed algorithm for finding the strongly connected components of a directed graph and then show that our algorithm is message-efficient. We assume that the reader is familiar with the notion of depth-first search in directed graphs. Suppose that a depth-first search is performed on a directed graph and that the depth-first numbers are assigned to each vertex in the order in which they are visited. Let DFN(v) denote the depth-first number of vertex v. The *root* of a strongly connected component is that vertex which has the least DFN among all vertices belonging to the component. For any vertex v, we define LOWLINK(v) as min($\{$DFN(v)$\} \cup \{$DFN(w) | there is a cross edge or a back edge from a descendant of v to w, and the root of the strongly connected component containing w is an ancestor of v $\}$). If the LOWLINK of every vertex is known, then the roots of the strongly connected components of the graph can be easily identified using the following lemma.

Lemma 1. A vertex v is a root of a strongly connected component if and only if LOWLINK(v) = v.□

Let the number of strongly connected components in G be k. Also let $r_1, \cdots, r_k$ be the roots of the strongly connected components $C_1, \cdots, C_k$ respectively such that the depth-first search of r_i terminated before that of r_j for $i < j$ and $1 \le i,j \le k$. It is possible to identify all members of a strongly connected component using the following lemma.

Lemma 2. For each i, $1 \le i \le k$, C_i consists of those vertices which are descendants of r_i but are in none of $C_1, \cdots, C_{i-1}$.□

We now consider the problem of assigning LOWLINK values to the vertices of the graph distributively. It is assumed that the leader, denoted by s, initiates the process of determining the strong components of the system. For the sake of simplicity, a path is assumed to exist from s to all other nodes in the network. Later, this condition will be relaxed and it will be shown how the algorithm can be extended to any general distributed system.

The technique is to perform a depth-first search of the graph and construct the strong components as the algorithm progresses. Initially, before performing the depth-first search, a spanning tree of the graph is constructed. This tree rooted at s, will be denoted by T_s. Each node maintains local variables NEW and AVAILABLE that are initialized to true. A true value for NEW indicates that the node has not yet been visited during the depth-first search while a true value for AVAILABLE denotes the fact that the node has not yet become part of any strong component.

After initialization, the algorithm to identify the strong components is initiated by s. This algorithm, called SEARCH, is shown in figure 1. It is built on top of a depth-first search initiated from s. As part of the initialization, s sends a message to itself to initiate the depth-first search. Apart from the variables described above, each node also maintains the following variables: NEXTDFN, DFN, LOWLINK, HISAVAILABLE and HISNUM. NEXTDFN is the value for the depth-first number to be assigned to the new node that is to be visited next. The variables DFN and LOWLINK are the depth-first number and LOWLINK values of the node itself. During the course of the algorithm, a node gets the values of AVAILABLE and LOWLINK or DFN of a neighbor to whom it had sent a message. These are stored in HISAVAILABLE and HISNUM respectively.

The algorithm uses the following types of messages: DFS, VISITED, UNVISITED, JOIN and ACK. A DFS message informs the receiver that it must participate in the depth-first search. Thus on receipt of a DFS message, a node v checks the value of NEW to see if it has been visited; if it is not visited, it continues the depth-first search from v by sending DFS messages to its successors in turn. After performing the depth-first search, the node informs the completion to its initiator by an UNVISITED message. However, a node that has a false value for NEW immediately replies to a DFS message with a VISITED message. Depending on the reply to a DFS message, a node adjusts its own LOWLINK value.

After completing its depth-first search, a node checks to see if it qualifies to be the root of a strongly-connected component. If so, it sends a JOIN message to itself and starts executing the algorithm STRONGCOMPONENT. This procedure causes each vertex that meets the requirement of lemma 2 to be identified. Consider a node v that receives a JOIN message inviting it to join the strong component C_i with root r_i. Assume that this message is received from node u. If AVAILABLE is true, then v joins C_i by setting the variable COMPONENT to the depth-first number of r_i. A message is sent to each of its successors inviting them to join C_i. It then waits until an ACK message is received from all of its successors. Then an acknowledgement is sent to u. Note that if v has already joined a strong component (C_i or any other), then it simply sends an acknowledgement to u without alerting any of its successors.

The General Case

In general, the leader will not have paths to all other vertices in the system. To tackle this problem, the leader constructs a spanning tree of G (ignoring the directed edges) and each node initializes local variables NEW and AVAILABLE to true. Let G_s denote the subgraph of G formed by all vertices that can be reached from s. A spanning tree of G_s, denoted by T_s, is constructed. The idea is to determine the strong components of G_s and then initiate the strong component identification algorithms at nodes not in G_s by traversing T_s in post-order fashion. Each node u in G_s instructs all nodes v that are predecessors of u to initiate the algorithm at v. If v has NEW and AVAILABLE true, it constructs the strong components of G_v, constructs T_v and recursively does a post-order traversal of T_v. The message is acknowledged and after initiating all its predecessors u reports to its parent in T_s. The details of the algorithm are given in figures 3 and 4.

Theorem 3. There exists an algorithm to determine the strong components of a distributed system using O(|E|) messages.

Proof. (Sketch). It is not difficult to check that by following the algorithms in figures 3 and 4 the algorithm to find the strong components will be executed at all nodes. Moreover, by performing a depth-first search, each node finds the LOWLINK values correctly using the algorithm in figure 3. The algorithm in figure 2 identifies exactly the members of a strong component. These facts imply that the algorithm indeed finds the strong components.

It may be noted that in the algorithms of figures 3 and 4, a constant number of messages are transmitted over any directed edge of the graph. Algorithms in figures 1 and 2 are executed for vertex disjoint components of the graph. Also it is a depth-first search algorithm combined with local processing. This needs messages proportional to the number of directed edges in the component. Thus the total number of messages is O(|E|). □

```
receive(u, v, DFS, NEXTDFN);
if NEW = true then
begin
          NEW ← false;
          DFN ← NEXTDFN;
          NEXTDFN ← NEXTDFN + 1;
          LOWLINK ← DFN;
          for each successor w of v do
          begin
                    send(v, w, DFS, NEXTDFN);
                    receive(w, v, type, HISAVAILABLE, HISNUM);
                    if type = UNVISITED or HISAVAILABLE = true then
                              LOWLINK ← min(LOWLINK, HISNUM)
          end;
          if LOWLINK = DFN then
          begin
                    send(v, v, JOIN, DFN);
                    STRONGCOMPONENT; (see figure 2)
                    send(v, u, UNVISITED, AVAILABLE, DFN)
          end
          else
                    send(v, u, UNVISITED, AVAILABLE, LOWLINK);
end
else
          if AVAILABLE = true then
                    send(v, u, VISITED, AVAILABLE, LOWLINK)
          else
                    send(v, u, VISITED, AVAILABLE, DFN)
```

Figure 1. Algorithm SEARCH

```
receive($u$, $v$, JOIN, ROOTID);
if AVAILABLE = true then
begin
    COMPONENT ← ROOTID;
    AVAILABLE ← false;
    for each successor $w$ of $v$ do
    begin
        send($v$, $w$, JOIN, ROOTID);
        receive($w$, $v$, ACK);
    end;
end;
send($v$, $u$, ACK);
```

Figure 2. Algorithm STRONGCOMPONENT

```
for each child $w$ of $u$ in $T_s$ do
    recursively execute the algorithm for $w$;
for each predecessor $v$ of $u$ do
begin
    send($u$, $v$, INITIATE);
    receive($v$, $u$, ACK);
end;
```

Figure 3. Post-order traversal of T_s

```
receive($u$, $v$, INITIATE);
if NEW = true and AVAILABLE = true then
begin
    find the strong components of $G_v$
    initiate the post-order traversal of $T_v$ in figure 3
end;
send($v$, $u$, ACK);
```

Figure 4. Response to the INITIATE message.

4. KNOT DETECTION

We now consider an application of strongly connected components. The problem of finding *knots* in networks is important in deadlock detection as shown by Chang [2]. A vertex v_i in a directed graph G is said to be a *knot* if for every vertex v_j that is reachable by a directed path from v_i, v_i is also reachable from v_j by a directed path. Chandy and Misra [6] present an elegant distributed algorithm to detect if a given node v_i is a knot. The algorithm searches the graph using the forward edges and finds all of the nodes reachable from the given vertex v_i. Similarly, all of the nodes from which v_i can be reached are also found by traversing the edges in the reverse direction starting from v_i. For each vertex v_j that can be reached from v_i, we can check if v_i can be reached from v_j. These results at the nodes are combined to check if v_i is a knot. It is easy to verify that the message complexity of the algorithm of [6] is $O(|E|)$. If all of the knots in the graph are to be identified, then running their algorithm at each node is very expensive as the message complexity becomes $O(|V||E|)$. We now show that $O(|E|)$ messages are sufficient to identify *every* knot in the system. Thus, our algorithm achieves an improvement of $O(|V|)$ in the number of messages.

Let $C_1, ..., C_k$ be the strongly connected components of the graph. For each component C_i, *an outgoing edge* is any edge (u, v) such that $u \varepsilon C_i$ and $v \varepsilon C_j$, $j \neq i$. In other words, an outgoing edge of a component is any edge that connects a node in that component to a node in another component. There is a close relationship between the strongly connected components of the graph and the knots as shown by the following theorems.

Theorem 4. A node $u \varepsilon C_i$ is a knot if and only if v is also a knot for all $v \varepsilon C_i$.

Proof. (sketch) It is easy to show that the set of nodes that can be reached from u is the same for all $u \varepsilon C_i$ as a directed path exists between every $u, v \varepsilon C_i$ and hence the proof follows. □

Theorem 5. The nodes in C_i are knots if and only if C_i has no outgoing edges.

Proof. (sketch) If part: Let u be a node in C_i. If C_i has no outgoing edges, then the set of nodes that can be reached from u is a subset of the set of nodes from which u can be reached.

Only if part: This part can be proved by contradiction using the definition of strongly connected components and the definition of a knot.□

To find all of the knots in the graph, we proceed as follows. First, the strongly connected components of the graph $C_1, ..., C_k$ are found using the algorithm presented earlier. For each C_i, a spanning tree rooted at r_i, the root of the strongly connected component C_i is also available. To check if there exists an outgoing edge for C_i, the following convergecast procedure can be used. Each leaf node checks if all of its successors are in the same strongly connected component by sending an *inquire* message and receiving a reply. The reply contains the name of the connected component to which the sender (of the reply) belongs. If all of the successors are in the same component, then the leaf node sets a local variable *outgoing* to false, and to true otherwise. It then sends a message containing the value of the local variable *outgoing* to its parent. Each interior node sets *outgoing* to false if there is no successor that is in a different component, and to true otherwise (this step is similar to that of the leaf nodes) and waits for a message containing true/false value from each child. It then computes the boolean *or* of the values it received from its children and its local variable *outgoing*, and sends the result to its parent. The root of the component behaves like an interior node and declares whether or not an outgoing edge for that component exists. The result is sent to all of the nodes in the component. By theorems 4 and 5, every node easily infers if it is a knot.

The message complexity can be analyzed as follows. The strongly connected components can be found using $O(|E|)$ messages using the distributed algorithm presented in section 3. The total number of *inquire* and *reply* messages is also $O(|E|)$ as exactly one such message is sent on each edge in either direction. For each component C_i, checking if an outgoing edge exists can be done using $O(|V_i|)$ messages and the result can be broadcast using $O(|V_i|)$ messages. Thus, the total number of messages used is $O(|E|)$.

Theorem 6. There exists an algorithm to detect *all of the knots* in a distributed system using $O(|E|)$ messages.☐

5. CONCLUSIONS

In this paper, we considered the problem of finding the strongly connected components of a directed graph. A message-efficient distributed algorithm was presented that uses $O(|E|)$ messages. We find that the all-knot detection problem can be solved using the strongly connected components using only $O(|V|)$ additional messages. The resulting algorithm finds all of the knots in a graph using only $O(|E|)$ messages. In contrast, the algorithm of [6] uses $O(|E|)$ messages to detect if a single node is a knot, and $O(|V|.|E|)$ messages are used if the knots are to be checked for at all nodes. Strongly connected components can also be used to find exactly those processors that are deadlocked.

We assumed that a bidirectional link exists between u and v for each directed edge (u,v). This assumption is very reasonable in many applications. If this is not the case, then the algorithm of Gafni and Afek [5] can be used to traverse the nodes of the system in a depth first manner, and then the strongly connected components can be found. However, the message complexity of the resulting algorithm is $O(|V||E|)$. For undirected graphs, the distributed algorithm of Chang [3] can be used to find the biconnected components.

References

1. Aho, A, Hopcroft, J, and Ullman, U., *The design and analysis of computer algorithms*, Addison Wesley, 1974.
2. Chang, E., ''Decentralized deadlock detection in distributed systems,'' *Technical report*, University of Victoria, B.C., Canada.
3. Chang, E., ''Echo algorithms: Depth parallel operations on general graphs,'' *IEEE Transactions on Software Engineering*, vol. SE-8, no. 4, pp. 391-401, 1982.
4. Gallager, R. G., Humblet, P.A., and Spira, P.M., ''A distributed algorithm for minimum weight spanning trees,'' *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66-77, 1983.
5. Gafni, E. and Afek, Y., ''Election and traversal in unidirectional networks,'' *Proceedings of the Third Annual Symposium on Principles of Distributed Computing*, pp. 190-198, 1984.
6. Misra, J. and Chandy, K.M., ''A distributed graph algorithm: knot detection,'' *ACM TOLPAS*, vol. 4, no. 4, pp. 678-686, 1982.

IMPROVING PETERSON'S CONSTRUCTION OF 1-WRITER n-READER MULTIVALUED ATOMIC REGISTER

K. VIDYASANKAR
Department of Computer Science
Memorial University of Newfoundland
St. John's, Newfoundland
Canada A1C 5S7

Abstract: A new construction improving Peterson's construction[ACM TOPLAS, Vol. 5(1), pp. 46–55, 1983 : Solution 3 (Main Solution)] of 1-writer n-reader multivalued atomic register is given in this paper. In the new construction both the read and the write operations do less work, and the "control structure" is simpler than in the original one.

A *shared register* (or simply a *register*) is an abstraction of asynchronous interprocess (persistent) communication, where the senders and receivers are called the *writers* and *readers*, respectively, and the state of the communication medium is the *value* of the register. A writer *writes*, that is, puts a value in the register, and a reader *reads*, that is, reports a value from the domain of the register. Both the read and the write operations are assumed to be *nonatomic.* A precedence relation (as in [1]) is defined on the set of operation executions: for operations a and b, *a precedes b*, denoted $a \rightarrow b$, if the execution of a finishes before that of b starts; a and b *overlap* if $\neg(a \rightarrow b \text{ or } b \rightarrow a)$. Asynchronous communication implies some operations overlap. In this paper we are concerned with 1-writer register. Then a write does not overlap any other write, that is, all the writes are totally ordered.

Registers have been classified according to their output characteristics in the following three ways [4].

1. A *safe* register is one in which a read not overlapping any write returns the most recently written value. A read that overlaps a write may return *any* value from the domain of the register, not necessarily the old value (that is, the value before the overlapping write) or the new one (the one written by the overlapping write).
2. A *regular* register is a safe register in which a read that overlaps a write obtains either the old value or the new value.
3. An *atomic* register is a safe register in which the reads and writes behave as if they occur in some total order which is an extension of the precedence relation.

Proposition 1[1,4]: A 1-writer n-reader atomic register is a regular one with the additional property that two successive reads that overlap the same write cannot obtain the new and then the old value.□

A register is *boolean* or *multivalued* depending upon whether it can hold only boolean or any number of desired values.

A 1-writer 1-reader boolean safe register is the easiest one that can be implemented with current hardware technology. (In a 1-reader register a read does not overlap another read.) These can then be used as building blocks to construct other registers. Several such constructions are given in [1–8]. This paper concerns with Peterson's construction[6: Solution 3 (Main Solution)] of 1-writer n-reader multivalued atomic register using $n+2$ multivalued safe registers, called *copy registers* and some boolean atomic registers, called *control registers.*

Peterson's construction is given in Figure 1 and will be denoted **P** in this paper. Here the writer writes new values in some copy registers, and each reader reads values from some copy registers and returns one of the values. The control registers "control" which copy registers are written in a write operation and which copy register value is returned in a read operation. Peterson has shown in [6] that $n+2$ copy registers are necessary and sufficient (where no reader nor the writer ever waits).

In this paper we give a new construction, given in Figure 2 and denoted **Q**, which is an improvement over Peterson's construction in the following respects:

- in **Q** a new value is written in at most $n+1$ copy registers, whereas in **P** it may be written in $n+2$ copy registers;
- in **Q** a read operation reads at most two copy registers, whereas in **P** at least two and upto three copy registers may be read; and
- the "control structure" in **Q** is simpler than in **P**.

In both the constructions, two of the copy registers are *main buffers* and the remaining are *copy buffers.* In **P** each write operation writes the new value in both the main buffers, whereas in **Q** it is written in only one of them. *The two main buffers are used alternately for writing successive new values.* This strategy of writing alternately appears to be the main reason for the simplicity of **Q** over **P**.

In the following we give a complete description of **Q**. In particular, the assignment statements are expanded so that each statement involves at most one shared register.

Detailed Description of Construction Q: The 1-writer n-reader multivalued atomic register R is composed of the following 1-writer registers: n-reader boolean atomic register

c, 1-reader boolean atomic registers $reading[1..n]$ and $writing[1..n]$, n-reader multivalued safe registers $buff_0$ and $buff_1$, and 1-reader multivalued safe registers $copybuff[1..n]$. The writer uses local (not shared) boolean variables cl and $wl[1..n]$. It is assumed that cl is initialized to either 0 or 1, and that, for each j, $reading[j]$, $writing[j]$ and $wl[j]$ are initialized to the same value, either 0 or 1. It is also assumed that the first operation of R is an initializing write operation that does not overlap with any read operation.

Algorithm for the ith reader:

```
read $k_1$ from writing[i];
write not $k_1$ in reading[i];
read k from c;
read $buff_k$;
read $k_2$ from writing[i];
if $k_2 = k_1$ (* $reading[i] \neq writing[i]$ *) then
    return $buff_k$ value
    else return copybuff[i] value;
```

Algorithm for the writer:

```
cl :=not cl;
write $buff_{cl}$;
write cl in c;
for $j := 1$ step 1 until n do
    begin
    read $k_1$ from reading[j];
    if $k_1 \neq wl[j]$ (* $reading[j] \neq writing[j]$ *) then
        begin
        write copybuff[j];
        write $k_1$ in writing[j] (* $writing[j] := reading[j]$ *);
        $wl[j] := k_1$
        end
    end; □
```

The *main buffers* are $buff_0$ and $buff_1$, and the *copy buffers* are $copybuff[1..n]$. The main buffers are used alternately for writing successive new values. Immediately after writing new value in a main buffer, the writer records that buffer index in c. Then the writer checks for each j whether a read operation of the jth reader overlapped the current write operation. This is done using a pair of boolean atomic registers $reading[j]$ and $writing[j]$. The reader sets these values different, by assigning the complement of $writing[j]$ to $reading[j]$, at the beginning of each read operation. Hence if the values are different when the writer checks them, the writer writes new value in *copybuff*[j] also and then sets the above values the same, by assigning the *reading*[j] value to *writing*[j].

A read operation of the jth reader, after reading *writing*[j] and writing its complement in *reading*[j] as mentioned above, checks c to find out which main buffer was written most recently, and reads from that buffer. Then it reads *writing*[j] again and compares with *reading*[j]. If the two values continue to be different, it returns the value just read from a main buffer; otherwise it reads *copybuff*[j] and returns that value.

Proof of Construction Q: For brevity, we will denote the register *reading*[j] as rj and *writing*[j] as wj in the following. We will also denote $buff_k$ as b_k, for k equal to 0 or 1, and *copybuff*[j] as cbj.

The register rj is written by the jth reader and read by the writer, and wj is written by the writer and read by the jth reader. The following property can be verified easily from the construction.

Lemma 2: Each time the value written in wj is the complement of the previous value in wj.□

We now introduce some terminology. Let a–read–p (a–write–p) denote the operation a reading (writing) register p. We note that each read operation r of the jth reader reads wj twice; we denote the first read operation r–1read–wj and the second r–2read–wj.

For operations a and b executed in the same boolean atomic register q, we say $a \Rightarrow_q b$ if a precedes b in the total ordering imposed on the operations by the atomic register.

Let r be a read operation of the jth reader.

Lemma 3: (a) Let w be the first write operation, if any, such that w–write–wj occurs in between r–1read–wj and r–2read–wj, that is, r–1read–$wj \Rightarrow_{wj} w$–write–$wj \Rightarrow_{wj}$ r–2read–wj. The operation w–write–wj sets $wj = rj$, and this equality continues to hold until the execution of r is complete.

(b) There is at most one write operation w such that w–write–wj occurs in between r–1read–wj and r–2read–wj.

(c) The operation r returns a main buffer value iff there is no write operation w such that w–write–wj occurs in between r–1read–wj and r–2read–wj; if there is such a w, then r returns cbj value which was written by w.

Proof: For part (a), without loss of generality assume that w writes 0 in wj. Then, by Lemma 2, the previous value of wj is 1. This implies r–1read–wj obtains 1 and therefore r–write–rj writes 0 in rj.

We note that, according to the construction, w first reads rj, and only if it finds $rj \neq wj$, it proceeds to write (cbj and then) wj. Thus w must have read 0 from rj; this is irrespective of whether w–read–$rj \Rightarrow_{rj} r$–write–rj or vice versa. Since the value written by r in rj is also 0, on execution of w–write–wj, both rj and wj have the same value, namely 0.

The value of rj remains 0 until r is completed. Suppose w is completed and the writer executes further write operations before r is completed. Each such operation w' will read 0 from rj and, since wj is also 0 and therefore $wj = rj$, will not write wj. That is, wj value is also unchanged and the equality $wj = rj$ holds until r is complete. This completes the proof of part (a). Parts (b) and (c) follow from part (a).□

Lemma 4: Suppose p is the (main or copy) buffer whose value was read and returned by r. Then when r was reading p, the writer was not writing in p. That is, "reading of p" did not overlap any "writing of p".

Proof: First suppose r returns a main buffer value. Without loss of generality, assume the buffer is b_0. Suppose there is a write operation w'' such that r reading b_0 overlaps w'' writing b_0.

Now r reads b_0 *after* reading 0 from c. Say w wrote this 0 in c. Now w writes in b_0 *before* writing in c. Hence w cannot be w''. Let w' be the immediate successor of w. Then w' will write in b_1, and not in b_0. Hence w' cannot be w'', and w'' must be a successor of w'.

From (i) r reads 0 from c, written by w and (ii) w' is a successor of w writing 1 in c, we have r–read–$c \Rightarrow_c w'$–write–c. Since r writes in rj before reading from c, and w' reads rj after writing in c, we have r–write–$rj \Rightarrow_{rj} w'$–read–rj. Now from (i) the assumption that w'' writing b_0 overlaps r reading b_0 and (ii) w'' is a successor of w', it follows that w' must finish execution before r–2read–wj. That is, if w' writes wj it should do so before r–2read–wj. Now w', on reading rj, will either find that $wj = rj$ (this would happen if w writes wj) or write in cbj and then write wj, setting $wj = rj$. In either case, by Lemma 3, the equality $wj = rj$ will hold until the execution of r is complete, and hence r will return cbj value. This contradicts the earlier assumption that r returns b_0 value.

Now suppose r returns cbj value. Let w be the write operation that wrote this value. Note that w writes cbj and then writes wj (setting $wj = rj$), whereas r finds this equality and then reads from cbj. Thus r reading cbj does not overlap w writing it. Furthermore since the equality holds until r is complete, no subsequent write operation will write cbj until r is complete, that is, until r finishes reading cbj.□

Lemma 4 assures that, though the copy registers are only safe, the read operations return “proper” values.

We now show the atomicity of R, after showing safeness and regularity. Safeness follows from the fact that a read operation of R that does not overlap with any write operation of R returns the value from the main buffer that was written by the most recent write operation.

Regularity: Let w be the last write operation that was completed before r started. Assume without loss of generality that w wrote in b_0.

First suppose r returned a main buffer value. If it is b_0, then clearly the value is that of w or of a successor that overlaps r. If it is b_1, then the value must be that of a successor that overlaps r which, after writing in b_1, writes 1 in c; r reads this 1 from c and decides to read from b_1.

Now suppose r returned copy buffer value. Assume r is of the jth reader. Then, by Lemma 3, some w writes wj in between r–1read–wj and r–2read–wj, and this is the same write operation that wrote the copy buffer value returned by r. That is, r returns the new value of an overlapping write.

Atomicity: We show that two successive reads of R overlapping the same write do not return the new and then the old values. Then from the regularity of R and Proposition 1, the atomicity follows.

Consider two reads r and r', such that $r \rightarrow r'$, overlapping the same write operation w. Assuming r returns the value written by w, we have to show that r' returns the value written by w or its successor. Assume r is an operation of the ith reader and r' is of the jth reader, where i and j need not be distinct.

Assume without loss of generality that w writes in the main buffer b_0. Since the writer writes in main buffer first, then writes in c and then in copy buffers, whether r returns the value (written by w) from main buffer or copy buffer it is clear that w writes 0 in c before the operation r is complete, that is, before r' starts. Therefore if r' returns a main buffer value, it is clearly written by w or a successor of w; and if r' returns copy buffer value, then by Lemma 3 it is the value written by a write operation w' that writes wj in between r'–1read–wj and r'–2read–wj. Since w is the first overlapping write of r', clearly w' is w or a successor of w.

Thus R is atomic.

The proof of the construction is complete.□

Acknowledgement : This research is supported in part by the Natural Sciences and Engineering Research Council of Canada Individual Operating Grant A-3182.

References

1. B. Awerbuch, L.M. Kirousis, E. Kranakis and P.M.B. Vitanyi, On Proving Register Atomicity, Report CS-R8707, Centre for Mathematics and Computer Science, Amsterdam, May 1987.
2. J.E. Burns and G.L. Peterson, Constructing Multi-reader Atomic Values From Nonatomic Values, Proc. PODC'87 Symp., pp. 222–231.
3. L.M. Kirousis, E. Kranakis and P.M.B. Vitanyi, Atomic Multireader Register, Report CS-R8722, Centre for Mathematics and Computer Science, Amsterdam, May 1987.
4. L. Lamport, On Interprocess Communication, Part I: Basic formalism, Part II: Algorithms, Distributed Computing, Vol. 1, pp. 77-101, 1986.
5. R. Newman-Wolfe, A Protocol for Wait-Free, Atomic, Multi-Reader Shared Variables, Proc. PODC'87 Symp., pp. 232–248.
6. G.L. Peterson, Concurrent Reading While Writing, ACM TOPLAS, Vol. 5(1), pp. 46–55, 1983.
7. A.K. Singh, J.H. Anderson and M.G. Gouda, The Elusive Atomic Register Revisited, Proc. PODC'87 Symp., pp. 206–221.
8. K. Vidyasankar, Converting Lamport's Regular Register to Atomic Register, Information Processing Letters, Vol. 28(6), pp. 287–290, 1988.

```
Algorithm for the ith reader:

reading[i] :=not writing[i];
sflag := wflag;
sswitch := switch;
read buff1;
sflag2 := wflag;
sswitch2 := switch;
read buff2;
bad1 := $sswitch \neq sswitch2$
    or sflag or sflag2;
if $reading[i] = writing[i]$ then
    return copybuff[i] value
elseif bad1 then return buff2 value
else return buff1 value;

Algorithm for the writer:

wflag := true;
write buff1;
switch := not switch;
wflag := false;
for j := 1 step 1 until n do
    if $reading[j] \neq writing[j]$ then
        begin
            write copybuff[j];
            writing[j] := reading[j]
        end;
write buff2;
```

Figure 1. P - Peterson's Construction

```
Algorithm for the ith reader:

reading[i] :=not writing[i];
read k from c;
read $buff_k$;
if $reading[i] \neq writing[i]$ then
    return $buff_k$ value
    else return copybuff[i] value;

Algorithm for the writer:

cl :=not cl;
write $buff_{cl}$;
write cl in c;
for j := 1 step 1 until n do
    if $reading[j] \neq writing[j]$ then
        begin
            write copybuff[j];
            writing[j] := reading[j]
        end;
```

Figure 2. Q - New Construction

AN ASYNCHRONOUS ALL PAIRS SHORTEST PATH ALGORITHM FOR SHARED MEMORY MULTIPROCESSORS†

AYDIN ÜRESİN and MICHEL DUBOIS

University of Southern California
Department of Electrical Engineering-Systems
Los Angeles, CA 90089-0781

Abstract

Three asychronous versions of Floyd's algorithm for shared memory multiprocessors are presented. Their difference from already existing counterparts is that they require no barrier synchronization. Therefore, they perform better for large numbers of processors. In the last part of the paper, the technique used for removing synchronization on the all pairs shortest path problem is identified and generalized.

1- INTRODUCTION

The class of shortest-path problems is one of the most important ones in the area of transportation and communication networks. Uniprocessor algorithms for this class are well studied and documented [1] , [4]. There have also been many parallel shortest-path algorithms studied in the literature [5], [7], [3], [9], [8]. Most of these algorithms assume an SIMD computational model which is not compatible with most existing machines. Some of them are indeed for loosely-coupled [6] or tightly-coupled [5] [9] multiprocessors, but they still have the drawback that the extensive use of some form of synchronization primitives degrades their performance, especially when the number of processors is large. At synchronization points processors are blocked: they either relinquish control to the operating system, spin on a single shared variable, or wait for access to a global queue.

In this paper, we propose a variation of Floyd's algorithm for the all-pairs shortest path problem. In the proposed scheme, no explicit synchronization is necessary. The processors are not blocked during the execution of the algorithm and there is no serial bottleneck. Cooperation among processors is restricted to shared data accesses. The cost of the scheme is the extra shared memory space requirement and the penalty due to accesses to this extra memory. Although we have only considered Floyd's algorithm, the technique we used for removing synchronization is more general and we will introduce it in Section 7.

2- SYNCHRONIZED PARALLEL FLOYD'S ALGORITHM

Given a directed graph in which each arc has a nonnegative cost, the problem of all-pairs shortest path is to find for each ordered pair of vertices (i, j) the smallest length of any path from i to j, where the length of a path is defined as the sum of the costs of the arcs on that path [1]. Floyd's Algorithm is considered the best for the solution of this problem and a parallel version of it is given in Figure 1 [5]. In this algorithm, n is the number of nodes in the graph and initially $A[i, j]$ is the cost

---

† This work was performed under NSF Grant No. CCR-8709997

```
for ROUND := 1 to n do
    forall $1 \le i, j \le n$ do
        $A[i,j] := \min\{A[i,j], A[i, ROUND] + A[ROUND, j]\}$
```

Figure 1: Synchronized parallel Floyd's algorithm

of the arc from i to j for all $i \neq j$; if there is no arc from i to j, $A[i,j]$ is set to ∞. Each diagonal element, $A[i,i]$, is set to 0. The keyword **forall** indicates that the $A[i,j]$'s are computed in parallel. To distinguish the shared writable variables from the other ones we adopt the convention of using uppercase names for such variables $(A, ROUND)$. The read/only shared variables (such as n) can be considered as local variables, because each processor can keep a copy of these in its local store and can efficiently access them. In the synchronized Floyd's algorithm, at the end of the k-th iteration (round), for all i, j : (i) $A[i,j]$ is the length of a path from i to j , (ii) $A[i,j]$ is not greater than the length of any path from i to j passing through nodes labelled k or less. Therefore, after the execution of the loop, matrix A contains the lengths of the mimimum paths for each ordered pair.

In the synchronized parallel version of Floyd's algorithm the computations of all components of A must be synchronized after each iteration. Therefore, a barrier synchronization is needed after each iteration of the loop. A *barrier* is a logical point in the control flow of an algorithm at which *all* the processors must arrive before any of them are allowed to proceed further, and it is the most deleterious form of synchronization [2]. Let p be the number of processors such that $p \le n^2$. If we ignore the execution of the barrier, each round of the algorithm in Figure 1 takes $O(n^2 p^{-1})$ time. However, in [2] it has been shown that the barrier introduces significant overhead and a straightforward implementation of it, that uses a single semaphore, takes $O(p\, t'(p))$ time where $t'(p)$ is the time complexity for executing the semaphore. A faster scheme is also given in [2] which uses $O(p \log p)$ memory and takes $O(t'(p) \log p)$ time. Here, we assume the second implementation without further elaborating on its complexity. As a result, the synchronized parallel Floyd's algorithm takes $O(n\,(n^2 p^{-1} + t'(p) \log p))$ time. When p is $O(n^2)$ then the second term dominates and the complexity becomes $O(n\, t'(n^2) \log n)$.

3- SIMPLE ASYNCHRONOUS FLOYD'S ALGORITHM

To remove the barrier, we first observe that the single variable $ROUND$ in the above algorithm has the role of keeping track of the "progress" of the data A towards the solution. After all the components of A are updated once, one unit of progress has been made and the next "round" of computation is entered. In the first asynchronous version of the algorithm which is given in Figure 2, on the other hand, each processor only keeps track of the progress of the component it is assigned to. Therefore, instead of only one $ROUND$ number for the whole data, we have a $ROUND$ number for each individual component. The idea is as follows. Suppose that at a certain instance of the computations the component $A[i,j]$ is in the k-th round ($k = ROUND[i,j]$). After $A[i,j]$ is updated in this round, there is a unit progress in $A[i,j]$ towards the solution if its predecessors are at least in the same round. In this case, $ROUND[i,j]$

```
      $task(i,j)$;
          begin {$task$}
S1:           $k\_old := ROUND[i,j]$ ;
S2:           $k := k\_old + 1$ ;
S3:           $ki := ROUND[i,k]$ ; $kj := ROUND[k,j]$ ;
S4:           $A[i,j] := \min\{A[i,j]\ ,\ A[i,k] + A[k,j]\}$ ;
S5:           if ($k\_old \leq ki$) and ($k\_old \leq kj$) then
                  begin
                      $ROUND[i,j] := k$ ;
                      if ($k = n$) then terminate
                  end ;
              return
          end {$task$}
```

Figure 2: Simple asynchronous Floyd's algorithm

can be incremented. By predecessors, we mean the components on which $A[i,j]$ is dependent in the k-th round, i.e., $A[i,j], A[i,k+1], A[k+1,j]$.

Before further elaborating on the above idea, we need to describe the model of computations. Figure 2 shows the program of the task that corresponds to a single update of the component $A[i,j]$. In our point of view, a task is an indivisible unit of execution that is held by the same processor from its start until the statement **return** is reached. There is a pool of tasks in the system: initially n^2 of them, one for each component. There are also a number of processors, and each available processor selects a task from the pool and executes it until the **return** statement. At this point, it returns the task to the pool and makes another task selection, or possibly continues with the same task. The execution of the **terminate** statement in $task(i,j)$ discards this task from the system for good, because $A[i,j]$ has reached the solution. It should also be mentioned that $ROUND[i,j]$ is initially assigned to 0, for all i,j.

The implementation of the task allocation policy that determines the selection rule is not made explicit here since a variety of schemes may be adopted. We only make two assumptions on the allocation. The first one is that the processors have exclusive hold of the tasks, i.e., at a given time instance $task(i,j)$ can be executed by at most one processor. This restriction is natural when each task is assigned statically to a fixed processor, throughout the computation. Static allocation has the advantage that it does not require an explicit task pool in the system: each processor repeatedly executes the task in its set with no need to access a global pool. If dynamic allocation is adopted, the exclusive hold condition should be enforced by some means. Another condition is that the allocation of tasks to processors is fair. In other words, in a finite period of time all tasks execute their code at least once. We will refer to these assumptions as the *exclusive hold* and *fairness* conditions.

The proof of correctness of the above described algorithm is based on the truth of the following property throughout the execution of the algorithm. Let $ROUND_t[i,j]$ and $A_t[i,j]$ be the values of $ROUND[i,j]$ and $A[i,j]$ at time t.

Property 1 *$A_t[i,j]$ is the length of a path from i to j which is not greater than the* *length of any path passing through the nodes labelled $ROUND_t[i,j]$ or less.*

Thus, $A_t[i,j]$ is the length of a minimum path when $ROUND_t[i,j] = n$. The correctness immediately follows from the following facts :

(1) At any time instance t during the execution of the algorithm Property 1 holds for all i, j,

(2) $ROUND[i,j]$'s are nondecreasing,

(3) For all i, j and t there exists a finite period of time Δt such that $ROUND_{t+\Delta t}[i,j]$ is greater than $ROUND_t[i,j]$.

We can show (1) by induction. It is obvious that initially Property 1 holds for all i, j (Recall that initially $ROUND[i,j] = 0$ for all i, j). After the initial state, the only events that would affect the truth of (1) are the executions of statements S4 and S5 in Figure 2. Let us assume that Property 1 holds just before the execution of S4. Then, it will obviously be satisfied after the execution of S4, as well, because the new value of $A[i,j]$ will not be greater than the old value. Now, let the old value of A be A_o and consider the time instance just before the execution of S4. If $ki \geq k\_old$, then $A_o[i,k]$ is not greater than the length of any path passing through nodes labelled k_old or less. The reason is that by induction hypothesis this is true at the time when ki is fetched and even if $A[i,k]$ might have been changed since then, this change can only decrease $A[i,k]$ without affecting the truth of the above statement. Similarly, if $kj \geq k\_old$, then $A_o[k,j]$ is not greater than the length of any path passing through nodes labelled k_old or less. Therefore, $A_o[i,k] + A_o[k,j]$ is smaller than or equal to the length of any path passing through k once and not passing through nodes with a label larger than k. Consequently, $\min\{A_o[i,j], A_o[i,k] + A_o[k,j]\}$ is the length of a path which is not greater than the length of any path passing through nodes labelled k or less, and Property 1 holds just after the execution of S5.

(2) is obvious from statement S4 which is the only statement that updates $ROUND[i,j]$. (3) is not as obvious. Let l be the minimum of all the components of $ROUND$ at time t and i', j' be the coordinates of this minimum, i.e., $ROUND[i',j'] = l$. From the fairness assumption in a finite period of time $task(i',j')$ will execute its code. During this execution $k\_old = l$ and therefore the condition of S5 is satisfied and $ROUND[i',j']$ is incremented. This means that $\min\{ROUND[i,j]\}$ is increased in a finite period of time and all the components are increased in a finite period of time.

The time complexity of this asynchronous Floyd's algorithm generally depends on the task allocation scheme, which affects the number of executions of $task(x,y)$, although the number of useful executions of $task(x,y)$ in which $ROUND[x,y]$ is incremented is always n. However, unless the load balance is very poor so that the frequencies of execution are widely different among tasks, then we can easily assume that each task is executed $O(n)$ times and between two consecutive executions of a certain task, there are $O(n^2 p^{-1})$ task executions by a single processor where p is the number of processors which is less than n^2. If accessing a shared variable takes $O(t(p))$ time, then a single execution of a task takes $O(t(p))$ time. Also assume an allocation scheme that does not take more than $O(t(p))$ to initiate the next task. Then the time complexity of the algorithm is $O(n^3 p^{-1} t(p))$. In the case when we have sufficient

```
    task(x, y);
        begin {task}
S1:         $k\_old := ROUND[x,y]$ ;
S2:         $k := k\_old + 1$ ;
S3:         $zx := Col\_Partition(k)$ ; $zy := Row\_Partition(k)$ ;
S4:         $kx := ROUND[x, zy]$ ; $ky := ROUND[zx, y]$ ;
S5:         for $(i,j) \in s[x,y]$ do
                $A[i,j] := \min\{A[i,j], A[i,k] + A[k,j]\}$ ;
S6:         if $(k\_old \leq kx)$ and $(k\_old \leq ky)$ then
                begin
                    $ROUND[x,y] := k$ ;
                    if $(k = n)$ then terminate
                end
S7:         return
        end {task} ;
```

Figure 3: Asynchronous Floyd's algorithm for partitioned data

number of processors such that p is $O(n^2)$, the complexity will be $O(n\, t(n^2) \log n)$, which is less than the synchronized case. Note that the time complexity of executing a semaphore $(O(t'(p)))$ is at least as large as the time complexity of accessing a shared variable $(O(t(p)))$.

4- ASYNCHRONOUS FLOYD'S ALGORITHM FOR PARTITIONED DATA

The synchronized Floyd's algorithm and the asynchronous Floyd's algorithm in the previous section are the two extreme cases with respect to the size of the *ROUND* data. As mentioned before, in the synchronized case only one *ROUND* number is maintained for the whole matrix A and in the simple asynchronous case, there is a *ROUND* number for each component. Using an additional memory for each component of A may be considered a waste, especially when there are less processors available than the number of components and therefore partitioning is necessary. Consequently, in the case of partitioning, we can utilize a *ROUND* number for each partition, instead of each component, and the result is the algorithm in Figure 3.

Here, the rows and the columns of matrix A are partitioned and a column partition x together with a row partition y defines a submatrix of A, i.e., a rectangular domain of indices which is denoted by $s[x,y]$ (Figure 4). $task(x,y)$ corresponds to a single update of the whole domain $s[x,y]$ instead of a single component update. The column partition that contains the k-th column is denoted by $Col\_Partition(k)$ and similarly the row partition that contains the k-th row is denoted by $Row\_Partition(k)$. The predecence relationships which existed among the individual components in the previous algorithm are now among the partitions.

The relationship between A and *ROUND* is stated by Property 2 below, which is analogous to Property 1.

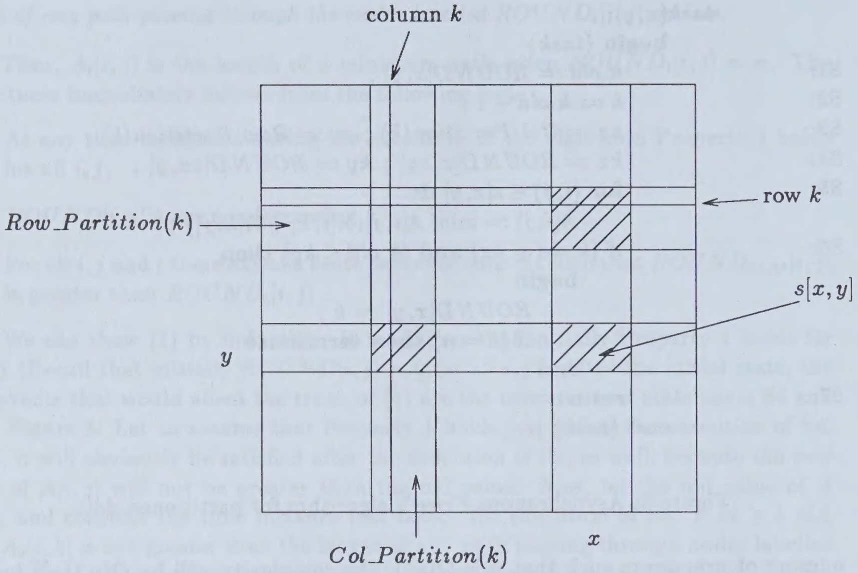


Figure 4: Partitioning A

Property 2 *$A_t[i,j]$ is the length of a path from i to j which is not greater than the length of any path passing through the nodes labelled $ROUND_t[x,y]$ or less, for all $(i,j) \in s[x,y]$.*

The argument to show that the elements of $ROUND$ eventually increase in a finite period of time till the solution, is the same as in the previous case; therefore, we will not repeat it here. The proof that Property 2 holds for all i,j and t is also similar to the proof of Property 1. We only have to notice that the only events that could affect the truth of Property 2 are the executions of S5 or S6. As a result, after $task(x,y)$ terminates the value of $ROUND[x,y]$ is n and from Property 2 $A[x,y]$ contains the solution.

As in the previous case, the time complexity of this version can be obtained easily as $O(n^3\, p^{-1}\, t(p))$. Although the complexity is the same, the size of $ROUND$, in this version, is smaller. Also, this version is faster due to the less frequent accesses to $ROUND$.

5- EARLY UPDATING OF *ROUND*

Notice that in the previous algorithm for partitioned data, $ROUND[x,y]$ is fetched at the beginning of $task(x,y)$ and it is updated at the end. This may cause the execution to slow down unnecessarily. To see this consider the following scenario. Consider the first executions of the tasks $task(x,y)$ and $task(x',y')$. $task(x,y)$ updates its partition and then updates $ROUND[x,y](=1)$. After this, it starts the second execution: it fetches $ROUND[x,y](=1)$ and later $ROUND[x,zy](=0)$ (at S4). Let $y' = zy$. If $task(x,y')$ is slower such that it updates $ROUND[x,zy]$ in the

```
        $task(x,y)$;
            begin {$task$}
S1:             $k\_old := ROUND[x,y]$ ;
S2:             $k := k\_old + 1$ ;
S3:             $k\_next := k + 1$ ;
S4:             $zx := Partition\_x(k)$ ; $zy := Partition\_y(k)$ ;
S5:             $kx := ROUND[x,zy]$ ; $ky := ROUND[zx,y]$ ;
S6:             for (($(i,j) \in s[x,y]$ and ($i = k\_next$ or $j = k\_next$)) do
                    $A[i,j] := \min\{A[i,j], A[i,k] + A[k,j]\}$ ;
S7:             if ($k\_old \leq kx$ and $k\_old \leq ky$) then
                    $ROUND[x,y] := k$ ;
S8:             for (($(i,j) \in s[x,y]$ and $i \neq k\_next$ and $j \neq k\_next$ do
                    $A[i,j] := \min\{A[i,j], A[i,k] + A[k,j]\}$ ;
S9:             if $ROUND[x,y] = n$ then terminate else return ;
            end {$task$}
```

Figure 5: Early updating of *ROUND*

first execution after $task(x,y)$ fetches it in the second execution, then in the second execution of $task(x,y)$ $ROUND[x,y]$ will not be incremented, because $task(x,y)$ will still see $ROUND[x,zy] = 0$. In general, the worst case is that a task increments its *ROUND* component in one of the two consecutive executions.

One of the ways to reduce the slowdown due to the above effect is to update $ROUND[x,y]$ as early as possible in $task(x,y)$. This is done in the algorithm in Figure 5.

The only difference between this version and the previous one is that $task(x,y)$ updates $ROUND[x,y]$ prematurely: $task(x,y)$ first updates the points in $s[x,y]$ either coordinate of which is $k\_next = ROUND_t[x,y] + 2$, then $ROUND[x,y]$ is updated before the remaining components in $s[x,y]$ are processed.

Property 3 *$A_t[i,j]$ is the length of a path which is not greater than the length of any path passing through the nodes labelled $ROUND_t[x,y]$ or less, for all i,j such that*

- *$(i,j) \in s[x,y]$, and*
- *$i = ROUND_t[x,y] + 1$ or $j = ROUND_t[x,y] + 1$.*

Since the only different part of the algorithm is between S7 and S9, the only possible cause of incorrect behavior of the algorithm may be this part. Particularly, the only reason the algorithm would not be correct could be as follows. $task(x,y)$ might update $ROUND[x,y]$ and another $task(\overline{x},\overline{y})$ might fetch this premature data before $task(x,y)$ finishes the execution of S8, and then it may use this premature data to produce incorrect data. On the other hand, $task(\overline{x},\overline{y})$ can only fetch $ROUND[x,y]$ at S5. Therefore, after S5 of $task(\overline{x},\overline{y})$, either $\overline{kx} = ROUND[x,y]$ or $\overline{ky} = ROUND[x,y]$. Without loss of generality, we assume the first case; from symmetry the argument for the second case is the same. $\overline{kx}$ is used only in S7 and since the difference between the premature and the old values of $ROUND[x,y]$ is 1, the correctness may be affected only when $\overline{k\_old} = \overline{kx}$, therefore $\overline{k} = ROUND[x,y] + 1$. On the other hand, the only elements of A that $task(\overline{x},\overline{y})$ uses as inputs are the ones either coordinate of which is $\overline{k} = ROUND[x,y]+1$. Property 3 states that Property 2 is satisfied for such elements. Therefore, the early update of $ROUND[x,y]$ does not make the algorithm incorrect.

Obviously, the complexity of this version of the asynchronous algorithm is the same as the previous one. However, since the processors make the $ROUND$ data available to other processors earlier than in the previous version, the components of $ROUND$ are incremented more frequently. Therefore, this algorithm executes faster.

6- PERFORMANCE CONSIDERATIONS

In the above sections we have shown that the asynchronous Floyd's algorithm is correct as long as the exclusive hold and fairness conditions are satisfied. We also mentioned that performance of the algorithm is directly related to the processor allocation scheme, because the more the load is balanced among the processors the less the total execution time is. Two possible allocation schemes are as follows:

1. Static allocation: the set of tasks is partitioned into disjoint subsets and each subset is preassigned to a processor. Each processor executes the tasks in its subset in a round-robin fashion.
2. Dynamic allocation: a given task is not executed by the same processor throughout its lifetime.

Static allocation has the advantage that it automatically satisfies the exclusive hold assumption, therefore no extra overhead is necessary to enforce it. Its disadvantage is that if the processors are not homogeneous or if the size of each partition cannot be made equal, performance may be degraded. Through dynamic allocation, on the other hand, the load balance can be tuned within the execution. By a balanced load we refer to the equal frequency of updates of A by each processor. However, exclusive hold condition must be enforced by some means. One way to accomplish this is to maintain a global queue of tasks and let the processors execute the tasks in this queue in a round-robin fashion. Accessing the global queue is the penalty for this scheme. A third allocation scheme that can be adopted is intermediate between the static and dynamic allocations. In this scheme most of the tasks are partitioned into equal sized subsets and the remaining tasks are allowed to float among the processors.

The most significant advantage of the asynchronous algorithm is that it has no barrier at which the processors have to wait idle. Barrier synchronization is an important cause of performance degradation, especially when the number of processors is high. On the other hand, the asynchronous algorithm requires extra memory space for $ROUND$ and extra overhead for accessing the elements of $ROUND$. Extra contention for accessing $ROUND$ can be reduced by placing the same components of $ROUND$ and A at consecutive locations in the shared memory and accessing them together. Also, there is no serial bottleneck such as accessing a single shared variable.

Another advantage of the asynchronous scheme is that by a minor addition the algorithm can detect an unbalanced load, because the components of $ROUND$ contain information about how many times individual components of A have been updated. A large difference between two components of $ROUND$ is an indication of an unbalanced load. When this situation occurs the system can be informed and proper tuning can be done.

```
for $k := 1$ to $m$ do
    forall $1 \le i \le n$ do
        $X[i] := Fi(X, k)$
```

Figure 6: A synchronized loop

7- GENERALIZATION OF THE TECHNIQUE

The above described technique of mapping a synchronized loop to an asynchronous one is in fact more general. The generalization follows from the observation that at a certain time instance, $ROUND[i,j]$ contains information about the range of values that $A[i,j]$ can take at this instance. In other words, $A[i,j]$ takes values from the domain that contains all the possible values of path lengths that are not greater than the length of any path passing through $1, 2, \ldots, ROUND[i,j]$. As $ROUND[i,j]$'s increase these domains shrink and finally when $ROUND[i,j] = n$ the domain contains a single element, the solution.

Suppose, we are given a synchronized loop as in Figure 6. Let there exist $m+1$ domains, $D(0), D(1), \ldots, D(m)$, such that the operator $F = F1 \times F2 \times \cdots \times Fn$ and $D(k)$ satisfies the following conditions:

1. $D(k) = D1(k) \times D2(k) \times \cdots \times Dn(k)$,
2. $X_0 \in D(0)$ (the initial value of X),
3. $D(k+1) \subseteq D(k)$,
4. $D(m) = \{\xi\}$,
5. $Y \in D(k) \Rightarrow F(Y,k) \in D(k+1)$,
6. $Y[i] \in Di(k) \Rightarrow Fi(Y,k) \in Di(k)$

From the above conditions it is easy to see that the result of the loop in Figure 6, i.e., the contents of X at the end of the execution is ξ. By using a similar argument as in Section 3 we can show that the result of the asynchronous loop in Figure 7 is

```
task(i) ;
    begin {task}
        $k\_old := \min\{ROUND[i]\}$ ;
        $X[i] := Fi(X, k\_old)$ ;
        $ROUND[i] := \max\{k\_old + 1, ROUND[i]\}$
        if ($ROUND[i] = n$) then terminate else return;
    end {task}
```

Figure 7: Asynchronous loop

$\xi[i]$. There are n tasks executing asynchronously and they satisfy exclusive hold and fairness assumptions. Notice that,

1. For all i, $X[i] \in Di(ROUND[i]) \Rightarrow X \in D(\min\{ROUND[i]\})$,
2. At all times, $X[i] \in Di(ROUND[i])$.

8- CONCLUSION

As we approach the era of massively parallel computing, overheads due to the interaction of processors such as synchronization and memory contention appear as important factors in the design and analysis of algorithms. Motivated by this fact, we have presented three versions of an asynchronous all pairs shortest path algorithm for shared-memory multiprocessor systems, free of barrier synchronization. Since, no barrier synchronization is involved and memory contention is minimized in these algorithms they perform better than previously announced ones, especially for a large number of processors. We showed the performance improvement by a simple complexity analysis. We finally generalized the idea used in the design of the asynchronous all pairs shortest path algorithms.

References

[1] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] Axelrod, T. S. Effects of synchronization barriers on multiprocessor performance. *Parallel Computing 3* (1986), 129–140.

[3] Dekel, E., D.Nassimi, and Sahni, S. Parallel matrix and graph algorithms. *SIAM J. Computing* (1981), 657–675.

[4] Deo, N., and C.Pang. Shortest path algorithms: taxonomy and annotation. *Networks* (1984), 275–323.

[5] Deo, N., Pang, C., and Lord, R. Two parallel algorithms for shortest path problems. In *IEEE International Conference on Parallel Processing* (1980), pp. 244–253.

[6] Jenq, J., and Sahni, S. All pairs shortest paths on a hypercube multiprocessor. In *International Conference on Parallel Processing* (1987), pp. 713–716.

[7] Paige, R., and Kruskal, C. Parallel algorithms for shortest path problems. In *IEEE International Conference on Parallel Processing* (1985), pp. 14–20.

[8] Quinn, M. J. *Designing Efficient Algorithms for Parallel Computers.* McGraw-Hill, 1987.

[9] Quinn, M. J., and Yoo, Y. B. Data structures for the efficient solution of the graph theoretic problems on tightly-coupled MIMD computers. In *IEEE International Conference on Parallel Processing* (1984), pp. 431–438.

TEST CASE CONSTRUCTION FOR NP-HARD PROBLEMS

LAURA A. SANCHIS[1]
Computer Science Department
University of Rochester
Rochester, New York 14627

Abstract: In evaluating the performance of approximation algorithms for NP-hard problems, it is often necessary to resort to empirical testing. In order to do such testing it is useful to have test instances of the problem for which the correct answer is known. We consider complexity issues arising in the efficient generation of sets of test instances with known answers. We also present techniques for generating such test instances for some NP-hard graph problems; we show that the test instances generated by these procedures are both hard and diverse.

1 Introduction

As a consequence of the apparent intractability of NP-complete problems, many of which are of great practical importance, much effort has been focused on designing and evaluating approximation algorithms for NP-complete and NP-hard problems. Approximation algorithms are usually based on heuristics and intuitions about the problem, but are not guaranteed to provide the correct answer in all instances, although they do run in polynomial time. For a number of such algorithms theoretical analysis has provided information about the worst case or expected performance of the approximation. Many times, however, empirical testing is the only practical means of determining the effectiveness of the algorithm, in particular when a specific application with specialized input instances is involved. In order to do such testing, it is useful to have test cases for which the correct answer is known. However, such test cases are not easy to obtain, since given an arbitrary instance of an NP-hard problem there is no known efficient procedure for determining the answer. In this paper we investigate some of the computational complexity issues involved in the efficient generation of test instances for NP-hard problems, and we also present useful test case construction techniques. We denote by a test instance of a problem a pair consisting of an instance of the problem together with its answer. A test set is a set of test instances. We present results concerning which kinds of test sets can or cannot be efficiently generated. We also present techniques that can be used to construct test instances for NP-hard problems in such a way that the test sets generated are both "diverse" and "hard", and we give examples of such construction methods.

Our model of test set generation is based on the following requirements. The test set generation procedure should be capable of outputting instances of the problem having a specified size (or other specified parameter values) and a given answer in an efficient manner (i.e. in polynomial time). Secondly, we require that the sets of instances generated be hard. By this is meant that unless P=NP there should be no polynomial time algorithm that can solve all of the instances generated by the procedure. Finally, since it is desired to generate test cases tailored to specific applications, we would like to be able to generate instances having a variety of desired characteristics. Our generation machine will be nondeterministic; that is, different instances may be outputted with different computations of the machine on the same input. We do not impose any particular distribution requirements on the generators. In many typical application situations, the distribution of actual problem instances for the algorithm being tested is not known: what is instead present is a usually ill-understood process that produces the instances of the problem that must be solved. Our goal in test case construction is to be able to generate instances resembling the real application problem instances (but for which the answer is known).

---

[1] AT&T Bell Labs Scholar

One question that naturally arises is the following: Is it possible to have a polynomial time test set generating procedure for an NP-complete or an NP-hard problem that for each given input size and answer, is capable of outputting all of the instances of the problem with that size and answer? It will be seen that for NP-complete problems this is not possible unless NP=co-NP, and the same is true for a large class of NP-hard optimization problems. However, it appears straightforward in many cases to find efficient procedures capable of generating all "yes" or positive instances of an NP-complete problem. We investigate whether or not this is true for all NP-complete or NP languages. It turns out that this question is hard to answer and has implications for other areas of complexity theory. An affirmative answer would imply that there are no sparse languages in NP-P, while a negative answer would imply that P $\neq$ NP. We define what it means for a set of instances of an NP-complete or NP-hard problem to be hard, and show how certain reductions can be used to show generated sets are hard. Any set of instances that is efficiently generable and that is proved to be hard using one of our reductions cannot be in P unless NP=co-NP. Finally, we present general techniques which appear to be useful in designing test case generators for graph problems. These techniques are applied to three NP-hard graph problems: the vertex cover, the graph colorability, and the dominating set problems. The generating algorithms for these problems produce test sets that can be proved to be hard as well as diverse.

Other work on test case construction for NP-hard problems can be found in [4] for network partitioning and in [9] and [8] for the symmetric traveling salesman problem. These papers deal with specific problems and do not consider the computational complexity issues involved in the problem of constructing test cases for NP-hard problems.

2 Generation Complexity

Definition 2.1 *Let* Π *be a problem and* $l_1, ..., l_k$ *be polynomial time computable functions whose domains consist of the set of instances for* Π *and whose ranges are the set of natural numbers and suppose* $l_j(I) \leq q(|I|)$ *for all* I, j *for some polynomial* q. *A test instance construction method (TICM) for* Π *(with respect to* $(l_1, ..., l_k)$*) is a nondeterministic polynomial time Turing machine* C *that given as input* $1^{v_1}\#...\#1^{v_k}$, *where* $v_1, ..., v_k$ *are nonnegative integers, and an answer* A, *outputs either an instance* I *of the problem having answer* A *such that for* $1 \leq j \leq k$, $l_j(I) = v_j$, *or the special symbol* Λ, *denoting that it cannot output any such instance. Denote by* $Gen(C, v_1, ..., v_k, A)$ *the set of all instances that may be generated by* C *on input* $1^{v_1}\#...\#1^{v_k}, A$; *by* $Gen(C, A)$ *the set of all instances generated by* C *with answer* A; *and by* $Gen(C)$ *all instances generated by* C.

In the above definition, when $k = 1$ and $l_1(x) = |x|$, we call the associated TICM a *length-based* TICM.

Definition 2.2 *A set of instances of a problem* Π *is* hard *with respect to* Π *if no polynomial time approximation algorithm for* Π *can give the correct answer for all of the instances in the set unless P=NP.*

We say that the set of instances generated by a TICM is *efficiently generable.* In this section we present complexity results about efficiently generable hard sets, and about which kinds of test sets can or cannot be efficiently generated. The proofs are omitted here but most may be found in [10] or [13].

We look first at decision problems. In this paper we assume that approximation algorithms for decision problems answer *yes*, *no*, or ? (*don't know*) (see [7]), always answering correctly in the first two cases. Both positive and negative test instances are required for testing different types of approximation algorithms for decision problems. Generation of positive instances really amounts to generating strings from the associated NP-complete language. For most NP languages we looked at it was not hard to design a TICM generating all positive instances of the associated problem. It turns out that determining whether or not this is true for all NP languages or even for all NP-complete languages would resolve some major open problem in complexity theory.

To clarify the discussion we define a polynomial time length-based language instance generator:

Definition 2.3 *A* polynomial time generator *(PTG) for a language L is a polynomial time nondeterministic Turing machine that on input 1^n either outputs some string of length n in L, or outputs Λ, indicating that no such string exists. Moreover, for each string x of length n in L, there exists some computation of the machine on input 1^n that outputs x.*

Observation 1 *An NP-complete problem has a length-based TICM generating all positive instances of the problem if and only if the associated NP-complete language has a PTG.*

It is fairly easy to see that any language that has a PTG must be in NP, since a PTG for a language L can be used as an NDTM for reconizing L in polynomial time. Answering the question of whether or not all NP languages have PTG's appears to be difficult. We have the following results.

Proposition 2.1 *All languages in P have PTG's iff all languages in NP have PTG's iff all NP-complete languages have PTG's.*

A language L is sparse if there exists a polynomial p such that the number of strings of length n in L is no more than $p(n)$. One of the currently open questions in complexity theory is whether there exist any sparse languages in NP-P.

Proposition 2.2 *If there is a language in NP that does not have a PTG then $NP \neq P$. If all languages in NP have PTG's then there are no sparse languages in NP-P.*

In regards to more general kinds of TICM's, we have the following:

Proposition 2.3 *If all NP-complete problems have length-based TICM's generating all positive instances of the problem, then all NP-complete problems have $(l_1, ..., l_k)$-TICM's generating all positive instances of the problem, for appropriate $l_1, ..., l_k$.*

The set of all instances of an NP-complete problem is of course hard according to Definition 2.2. In fact, even the set of all positive instances is hard (although not sufficiently diverse). If we have a TICM generating some but not all positive instances of an NP-complete problem, then the following proposition may be useful in showing that the instances generated are still hard.

Proposition 2.4 *Let Π be an NP-complete problem, and L be the associated NP-complete language. Let M be an NP-complete language, and let B be a set of instances of Π. Suppose there exists a polynomial time many-one reduction f from M to L such that if $x \in M$ then $f(x) \in B$. Then B is hard with respect to L.*

The situation for *no* instances is quite different from that for *yes* instances. For if there exists a TICM generating all *no* instances of an NP-complete problem, i.e. instances of the problem that are not in the associated NP-complete language, then this set would be in NP, i.e. the complement of an NP-complete language would be in NP, implying that NP=co-NP [3].

Proposition 2.5 *If Π is an NP-complete decision problem, and if there exists a TICM C for Π such that $Gen(C, no)$ consists of all instances of the problem for which the answer is* no, *then NP=co-NP.*

Corollary 2.1 *If Π is an NP-complete decision problem and there exists a TICM generating all instances of Π, then NP=co-NP.*

We turn now to NP-hard optimization problems. A typical approximation algorithm A for an NP-hard maximization (minimization) problem takes as input an instance I of the problem and outputs a result $A(I)$ which is no greater (smaller) than the optimal answer $OPT(I)$. Although in one sense it would suffice to have positive instances of the associated decision problem, i.e. instances I, k where $OPT(I) \geq k$ ($OPT(K) \leq k$), in order to test such algorithms, it is more useful to have instances of the form I, k where $OPT(I) = k$ in order to see how far the approximation algorithm is from optimal. The following results parallel some of the results for decision problems.

Proposition 2.6 *Let Π be an NP-hard optimization problem consisting of instances I with corresponding answers $OPT(I)$, and suppose that the language consisting of the tuples $(I, OPT(I))$ is not in NP unless NP=co-NP. Then if there exists a TICM generating all instances of Π, NP = co-NP.*

It is shown in [5] that a variety of NP-hard optimization problems, including but not restricted to those derived from strongly NP-complete problems, have the property that the set of tuples $(I, OPT(I))$ is not in NP unless NP=co-NP. By the above proposition these problems cannot have TICM's generating all instances unless NP=co-NP.

Proposition 2.7 *Let Π be an NP-hard maximization (minimization) problem and B a set of instances of Π. Let L consist of the language of tuples of the form (I, k) where I is an instance of Π and $OPT(I) \geq k$ ($OPT(I) \leq k$). If M is an NP-complete language and f is a polynomial time many-one reduction from M to L such that if $x \in M$ and $f(x) = (I_x, k_x)$ then $I_x \in B$, then B is hard with respect to Π.*

Proposition 2.8 *If $B = Gen(C)$ where C is a TICM for an NP-complete or NP-hard problem Π, and if B has been shown to be hard with respect to Π using either Proposition 2.4 or Proposition 2.7, then B cannot be in P unless NP=co-NP.*

A modification of a proof in [6] of the existence of complexity cores for languages not in P yields the following.

Proposition 2.9 *Let Π be an NP-complete decision problem and B a set of instances of Π that is hard with respect to Π. Then unless P=NP, there exists an infinite set $X \subseteq B$ such that for all polynomials p, any algorithm that solves Π exactly takes more than $p(|x|)$ time on all except a finite number of $x \in X$.*

3 Test Case Construction Techniques

3.1 Introduction

As seen in the last section, for most NP-hard problems it is unlikely that we will be able to find a TICM generating all instances of the problem. In this section we show that it is still possible, at least in some cases, to construct TICM's that generate hard and diverse sets of instances. We present efficient procedures for generating hard sets of test instances for the vertex cover problem, the graph colorability problem, and the dominating set problem. Each of these procedures is capable of generating graphs having any specified number of vertices, edges, and a given answer, whenever such graphs exist.

The design of the test case construction algorithm for each of these problems combines three separate but interdependent aspects. The first of these has to do with the construction of a graph to ensure that it will have a specific answer. In order to do this we start with certain subgraphs that when embedded in a larger graph will ensure that the "answer" (e.g. minimum vertex cover size) will be at least a certain value. We then add extra edges to hide these subgraphs, at the same time ensuring that the lower bound on the answer is actually attained by the final constructed graph. Some of these construction techniques were originally inspired by methods used in [4].

The second aspect has to do with ensuring that the generating procedure is capable of producing graphs with any given number of vertices, number of edges, and answer, whenever such graphs exist. To do this we first have to determine what the valid combinations of vertex count, edge count, and answer are for each specific problem. We thus need certain results from extremal graph theory, some of which we have taken from the literature and some of which are our own results. Using these results we customize the construction method described in the last paragraph to make sure that the procedure can generate a sufficient variety of graphs, and that it can produce graphs with given specific parameter values.

Finally, the third aspect consists of ensuring that the generated instances are hard. To do this we use reductions from NP-complete problems to our NP-hard problems, and if necessary modify and generalize these reductions in order to be able to apply Proposition 2.7 to the generated instances. These reductions can be further generalized so that a large variety of the graphs generated by the construction algorithms can be shown to be in the range of the reductions, and can thus be shown to constitute hard sets of instances for the problem.

Several reductions from the NP-complete problem 3-satisfiability will be mentioned and/or presented in this section. An instance of the 3-satisfiability problem consists of a set $U = \{u_1, ..., u_n\}$ of variables and a set $C = \{c_1, ..., c_m\}$ of clauses. Each clause consists of three literals; literals are of the form u_i or $\bar{u}_i$. The formula coded in C is satisfiable if and only if there exists a truth assignment t that assigns the value T (true) or F (false) to each variable in such a way that each clause c_j contains at least one literal assigned the value T (If t assigns the value T to u_i then it assigns the value F to $\bar{u}_i$, and viceversa). The 3-satisfiability problem consists of determining whether a given formula is satisfiable.

3.2 Vertex Cover

A vertex cover for a graph $G = (V, E)$ consists of a subset of the vertices $V' \subseteq V$ such that every edge of G is incident on at least one vertex of V'. The vertex cover problem consists of finding the cardinality of the smallest possible vertex cover for a given graph G. This problem is NP-hard ([3]).

Let S be a set of pairs of the form $\langle G, V' \rangle$ where $G = (V, E)$ is a graph and $V' \subseteq V$ is a minimum vertex cover set for G. Then other graphs with known minimum covers can be constructed as follows. Select any $\langle G_1, V_1' \rangle, \langle G_2, V_2' \rangle, ..., \langle G_k, V_k' \rangle$ from S and take their disjoint union; then add zero or more edges in such a way that each added edge is incident on some vertex from $\bigcup_{i=1}^{k} V_i'$. The generated graph will then have a minimum vertex cover consisting of $\bigcup_{i=1}^{k} V_i'$.

Denote by $gen(S)$ the set of graphs that can be generated from a set S as described above. Denote by $ver(G)$ the number of vertices of G, and by $vc(G)$ the size of a minimum vertex cover for G. There exists a graph with n vertices and minimum vertex cover size c if and only if $0 \leq c \leq n - 1$. Suppose it is desired to generate a graph with n vertices and minimum vertex cover size c. Then one should choose k graphs $G_1, G_2, ..., G_k$ from S such that $\sum_{i=1}^{k} ver(G_i) = n$ and $\sum_{i=1}^{k} vc(G_i) = c$. If S contains all cliques, this can be done by setting $k = n - c$, randomly choosing positive integers $n_1, n_2, ..., n_k$ such that $\sum_{i=1}^{k} n_i = n$, and letting G_i be a clique of n_i vertices, for each i. The desired graph is formed by taking the disjoint union of the G_i and adding extra edges as desired.

One can also control the number of edges in the generated graph by choosing appropriate cliques G_i. In order to produce a graph having m edges, one should choose the cliques in such a way that the total number of edges in their disjoint union is less than or equal to m (since extra edges can be added later). In order to do this we need to find out for given n, c, what the minimum number of edges is that can be obtained from a graph formed from the union of cliques as described above.

Proposition 3.1 *Let $n \geq 1, c \geq 0, m \geq 0$ be integers. There exists a graph with n vertices, m edges, and minimum vertex cover size c iff $0 \leq c \leq n-1$ and $h_1(n,c) \leq m \leq h_2(n,c)$ where h_1, h_2 are defined as follows. Given n, c, let $k = n - c$, and let b, r be the quotient and remainder when n is divided by k, i.e. $n = kb + r$ where $b \geq 0, 0 \leq r < k$. Then*

$$h_1(n,c) = r\binom{b+1}{2} + (k-r)\binom{b}{2}$$

$$h_2(n,c) = \binom{c}{2} + kc$$

Proof: As remarked above, we must have $0 \leq c \leq n-1$, and for any such n, c there exists a graph with n vertices and minimum vertex cover size c. Consider a graph with n vertices and minimum vertex cover size c. Every edge must be incident on at least one of the vertex cover vertices, hence all of the possible edges of the graph must be among the $\binom{c}{2}$ edges joining vertices in the cover, and the $c(n-c)$ edges joining a vertex in the cover to one not in the cover. So the graph can have at most $h_2(n,c)$ edges. Moreover a graph with this maximum number of edges does exist. The above argument also shows that for any $m_1 \leq m_2 \leq h_2(n,c)$, if there exists a graph with n vertices, minimum vertex cover size c, and m_1 edges, then there exists a graph with n vertices, minimum vertex cover size c, and m_2 edges. It only remains to show that the minimum number of edges a graph with n vertices and minimum vertex cover size c can have is $h_1(n,c)$. This follows from Turan's theorem [14], which can be stated to say that the only graph with n vertices, minimum vertex cover of size c, and $h_1(n,c)$ edges, is the graph consisting of r cliques of size $b+1$ and $(k-r)$ cliques of size b (k, b, r are defined in the statement of the proposition), while any graph of n vertices and minimum vertex cover of size c must have at least this many edges. ([15] expresses this in terms of maximum cliques, while a statement for maximum independent sets can be found in [2]). ■

The following proposition now follows from the above proposition and the fact that the lower bound on the number of edges given in that proposition can always be achieved by a disjoint union of cliques.

Proposition 3.2 *If S contains all cliques, then for all n, m, c such that there exists some graph with n vertices, m edges, and minimal vertex cover size c, gen(S) contains at least one such graph.*

It can also be shown that no polynomial time approximation algorithm for the vertex cover problem can provide the right answer for all instances in *gen*(S), for appropriate S.

Proposition 3.3 *Suppose S contains a clique of size 2 and a clique of size 3 (together with possibly other graphs). Then gen(S) is hard with respect to the vertex cover problem.*

This follows from a reduction found in [3] which reduces 3-satisfiability to the vertex cover problem. By observing that whenever a formula is satisfiable, the graph produced by the reduction is in *gen*(S), the result follows from Proposition 2.7. We can also remark that although the graphs produced by this reduction are all based on 2-cliques and 3-cliques, in the reduction each 3-clique may be replaced by a q-clique for any $q \geq 3$. This shows that the hardness of the generated instances is not restricted to those generated using only 2-cliques and 3-cliques.

3.3 Graph Colorability

A k-coloring for a graph $G = (V, E)$ is a function $f : V \rightarrow \{1, 2, ..., k\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$. The graph colorability problem consists of finding for a given graph

G the minimum k such that there exists a k-coloring for G. Such a k is called the chromatic number of G. The graph colorability problem is NP-hard [3].

The set of graphs having chromatic number k that will be generated by our method will be exactly the set of graphs that have chromatic number k and that contain a subgraph consisting of a k-clique. It is not hard to see that if a graph has n vertices, m edges, chromatic number k, and k-coloring function f such that f assigns the ith color to exactly n_i vertices, then $\binom{k}{2} \leq m \leq (\binom{n}{2} - \sum_{i=1}^{k} \binom{n_i}{2})$ (the lower bound can be proved by induction on k). In order to construct such a graph, first pick k vertices and form a k-clique. Then assign a distinct color to each of the vertices in the clique. We then assign colors to the $n - k$ remaining vertices in such a way that color i is assigned to exactly n_i of the n vertices. Finally, optionally add any edge joining any two vertices that have been assigned different colors, in order to obtain a total of m edges. It is clear that this procedure can be used to generate any graph with n vertices and chromatic number k that contains a k-clique as a subgraph. Also, this procedure is capable of generating some graph with n vertices, m edges and chromatic number k, whenever such a graph exists.

Finally, we claim that the set of graphs just described is hard. This follows from Proposition 2.7 and the fact that a polynomial time reduction exists from the 3-satisfiability problem to the colorability problem such that if a formula is satisfiable, then the graph constructed by the reduction belongs to the set of graphs generated by our procedure. This reduction can be found in [1]. The following corollary follows from Proposition 2.8.

Corollary 3.1 *The set of graphs for which the size of the maximum clique equals the chromatic number of the graph is not in P unless NP=co-NP.*

3.4 Dominating Set

A dominating set for a graph $G = (V, E)$ is a subset of vertices $V' \subseteq V$ such that for all $v \in V - V'$, there exists some $u \in V'$ for which $\{v, u\} \in E$. The domination number of a graph is the size of its smallest dominating set. The dominating set problem consists of finding the domination number for a graph. The dominating set problem is NP-hard [3].

It is possible to efficiently generate all graphs having domination number 2. More specifically, there exists a polynomial time nondeterministic generating procedure that takes as input the desired number of vertices and number of edges, and outputs (if possible) a graph with domination number 2 having these characteristics; this procedure generates all such graphs [12]. We will in this paper concentrate on the generation of connected graphs having domination number at least 3.

Definition 3.1 *A* dominating core graph *consists of a triple (G, x, y) where G is a graph and x, y are vertices of G (not necessarily distinct) such that x is adjacent to every other vertex of G.*

The construction method consists of selecting k dominating core graphs (G_1, x_1, y_1), ...,(G_k, x_k, y_k), taking the disjoint union of $G_1, ..., G_k$, selecting $\bigcup_{i=1}^{k}\{x_i\}$ as the minimum size dominating set for the graph to be constructed, and then adding extra edges in such a way that no extra edge is incident on any of the y_i-vertices. If this procedure is followed, the x_i will constitute a dominating set of size k for the resulting graph. Moreover, any dominating set must contain at least one vertex from each G_i, since y_i is not adjacent to any vertex outside of G_i. Therefore the domination number of the constructed graph is exactly k.

It can be shown that using dominating core graphs, graphs with any specified valid combination of number of vertices, number of edges, and domination number can be constructed. Before we can show this we need to determine what the valid combinations are.

The following lemmas and propositions answer this question.

Lemma 3.1 *A connected graph $G = (V, E)$ with n vertices has domination number no greater than $n/2$.*

Proposition 3.4 *Let G be a connected graph with n vertices and domination number d where $3 \leq d \leq n/2$. Then the number of edges of G is at most $\binom{n-d+1}{2}$. A graph having exactly this number of edges must be of the following form:*

(1) An $(n-d)$-clique, together with an independent set of size d, such that each of the vertices in the $n-d$-clique is adjacent to exactly one of the vertices in the independent set, and such that each of these d vertices has at least one vertex adjacent to it.

(2) For $d = 3$, G may consist of a clique of $n-5$ vertices, together with 5 vertices x_1, x_2, x_3, x_4, x_5, with edges $\{x_1, x_3\}, \{x_2, x_4\}, \{x_2, x_5\}$, such that every vertex in the $n-5$-clique is adjacent to x_4 and x_5, and in addition adjacent to either x_1 or x_3. Moreover, at least one of these vertices is adjacent to x_1 and at least one to x_3.

A proof of the above proposition may be found in [11].

Lemma 3.2 *If G is a connected graph with n vertices and domination number d, then G has at least $n-1$ edges. This bound is tight.*

By using the above proposition and lemmas, it is not hard to see the following.

Proposition 3.5 *There exists a connected graph with n vertices, m edges, and domination number $d \geq 3$ if and only if $d \leq n/2$ and $n - 1 \leq m \leq \binom{n-d+1}{2}$.*

The following procedure will generate a connected graph with n vertices, m edges, and domination number $d \geq 3$, for any valid n, m, and d.

Input: n, m, d where $n \geq 6$, $3 \leq d \leq n/2$, and $n - 1 \leq m \leq \binom{n-d+1}{2}$.

Algorithm: Start out by choosing d vertices $x_1, \ldots, x_d$ which will constitute a dominating set for the graph. Connect each of the other vertices to exactly one of the x_i with an edge, making sure that each x_i has at least one such vertex connected to it (this can be done since $d \leq n/2$). We thus have d components and $n - d$ edges. For each i, $1 \leq i \leq d$, choose any one vertex from the component containing x_i; call this vertex y_i. (The vertex y_i may or may not be equal to x_i). Now add $m' = m - n + d$ edges in such a way that no edge adjacent to a y_i vertex is adjacent to any vertex outside the ith component, and also in such a way that $d - 1$ of these edges serve to connect up the d components. This can be done since

$$d - 1 \leq m' \leq \binom{n-d+1}{2} - (n-d) = \binom{n-d}{2}$$

The resulting graph will have domination number d.

We now show that the set of graphs generated by the above algorithm is hard. Our proof is based on a reduction that is a combination of a reduction from 3-satisfiability to vertex cover, and a reduction of vertex cover to dominating set. The first is given in [3] while the second is suggested there as an exercise. Our reduction is both a simplification and generalization of this composed reduction.

Let k be an integer no smaller than 3. Let variables $U = \{u_1, u_1, \ldots, u_n\}$ and clauses $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of 3-satisfiability.

Corresponding to each variable u_i construct a graph A_i having at most k vertices, and containing three distinguished vertices $u_i, \bar{u}_i, x_i$. Both u_i and $\bar{u}_i$ should be adjacent to every other vertex in A_i. Corresponding to each clause c_j construct two graphs B_j and C_j, each having at most k vertices. B_j should contain three distinguished vertices d_j, e_j, f_j, and d_j

and e_j should each be adjacent to every other vertex in B_j. Likewise, C_j should contain three distinguished vertices g_j, h_j, i_j, and g_j and h_j should each be adjacent to every other vertex in C_j. Then connect these subgraphs as follows. For each clause c_j create three vertices $c_{j,1}, c_{j,2}, c_{j,3}$. If the literals in clause c_j are l_j, m_j, n_j (e.g. $l_j = u_i$ or $l_j = \bar{u}_i$ for some i), then we add three edges connecting $c_{j,1}$ to l_j, $c_{j,2}$ to m_j, and $c_{j,3}$ to n_j. In addition, we add edges connecting $c_{j,1}$ to d_j and to g_j, $c_{j,2}$ to e_j, and $c_{j,3}$ to h_j. Call the final constructed graph G. For fixed k the above construction can be carried out in polynomial time in the size of the original formula.

Note that G must have domination number at least $n + 2m$. This is because for each i, $1 \leq i \leq n$, vertex x_i is adjacent only to vertices in A_i, and therefore A_i must contain at least one dominating set vertex. Similarly, B_j and C_j must each contain at least one dominating set vertex.

Suppose that the formula is satisfiable. Then we can find a dominating set D for G as follows. Choose a satisfying assignment t for the formula. For each i, if t assigns the value T (true) to u_i, put vertex u_i in D, else put $\bar{u}_i$ in D. Consider clause c_j. Since t is a satisfying assignment, at least one of the literals l_j, m_j, n_j is set to true by t. If l_j is set to true, put e_j and h_j in D. Otherwise, if m_j is set to true, put d_j and h_j in D. Otherwise (n_j is set to true) put e_j and g_j in D. This makes for a total of $n + 2m$ vertices in D. Every vertex in each A_i, B_j, and C_j is clearly adjacent to one of the vertices in D. It remains only to make sure that for each j the vertices $c_{j,1}$,$c_{j,2}$, and $c_{j,3}$ are each adjacent to some vertex in D. By the manner in which the vertices in D were selected from B_j and C_j it is clear that at least two of $c_{j,1}$, $c_{j,2}$, $c_{j,3}$ are adjacent to some of these vertices in B_j and/or C_j. The third vertex is adjacent to some u_i or $\bar{u}_i$ which was set true by t and is therefore also in D. Thus D is a dominating set for G, and G has domination number $n + 2m$.

Suppose now that G has domination number $n + 2m$ and let D be a dominating set of size $n + 2m$ for G. We know that each A_i, each B_j, and each C_j must contain at least one vertex in D, and therefore they each contain exactly one vertex in D. Since among the vertices in A_i, only u_i and $\bar{u}_i$ are adjacent to vertices outside A_i, and since each of these vertices is adjacent to every other vertex in A_i, we can if necessary modify D so that either u_i or $\bar{u}_i$ is in D, while retaining the property that D is a dominating set for G. Likewise we can assume that either d_j or e_j is in D, and either g_j or h_j is in D, for each j. Define an assignment t for U as follows. If u_i is in D, set $t(u_i) = T$, otherwise set $t(u_i) = F$. To see that t is a satisfying assignment, consider any clause c_j. Since only two of d_j, e_j, g_j, h_j can be in D, at least one of the vertices $c_{j,1}, c_{j,2}, c_{j,3}$ is not adjacent to any dominating set vertex in B_j or C_j. Hence this vertex must be adjacent to some u_i or $\bar{u}_i$ vertex in D, implying that the corresponding literal in the clause c_j is set to true by t. So we have shown that t is a satisfying assignment and therefore the original formula was satisfiable.

It remains to show that if the formula is satisfiable, the constructed graph is in the set generated by our procedure. Suppose the formula is satisfiable, and let D be a dominating set for G consisting of exactly one vertex from each of A_i, B_j, and C_j. In fact, as shown above, we may assume that either u_i or $\bar{u}_i$ is in D, either d_j or e_j is in D, and that either g_j or h_j is in D. Define subgraphs $\tilde{A}_i$, $\tilde{B}_j$, and $\tilde{C}_j$ as follows. Each $\tilde{A}_i$, $\tilde{B}_j$, and $\tilde{C}_j$ will contain all of the vertices and edges in A_i,B_j, and C_j, respectively, with possibly other vertices from among the $c_{j,1}, c_{j,2}, c_{j,3}$, together with connecting edges. Each vertex w of the form $c_{j,1}$,$c_{j,2}$, or $c_{j,3}$ is adjacent to at least one vertex in D from either A_i, B_j, or C_j. Choose one such set that w is adjacent to and let this vertex and its connecting edge belong to corresponding $\tilde{A}_i$, $\tilde{B}_j$, or $\tilde{C}_j$. Now each of the sets$\tilde{A}_i$, $\tilde{B}_j$, and $\tilde{C}_j$ have a dominating set vertex that each of the other vertices in the set is adjacent to, and they also each have a vertex (x_i, f_j, or i_j) that is not adjacent to any vertex outside the set. So all the $\tilde{A}_j$, $\tilde{B}_j$, and $\tilde{C}_j$ are dominating core graphs and G is of the required form.

Acknowledgements

The author is grateful to Mark Fulk for his advice and support and many useful discussions, to Mandayam Srinivas for many discussions while these ideas were germinating, and to Joel Seiferas for useful comments and suggestions.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] Claude Berge. *Graphs.* North-Holland, 1985.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[4] Balakrishnan Krishnamurthy. Constructing test cases for partitioning heuristics. *IEEE Transactions on Computers*, C–36(9), September 1987.

[5] E.W. Leggett and Daniel J. Moore. Optimization problems and the polynomial hierarchy. *Theoretical Computer Science*, 15:279–289, 1981.

[6] Nancy Lynch. On reducibility to complex or sparse sets. *Journal of the Association for Computing Machinery*, 22(3):341–345, July 1975.

[7] Pekka Orponen, David A. Russo, and Uwe Schoning. Optimal approximations and polynomially levelable sets. *SIAM Journal of Computing*, 15(2):399–408, May 1986.

[8] Martha G. Pilcher and Ronald L. Rardin. Invariant problem statistics and generated data validation: Symmetric traveling salesman problems. Technical Report CC-87-16, Purdue University, Institute for Interdisciplinary Engineering Studies, April 1987.

[9] Martha G. Pilcher and Ronald L. Rardin. A random cut generator for symmetric traveling salesman problems with known optimal solutions. Technical Report CC-87-4, Purdue University, Institute for Interdisciplinary Engineering Studies, February 1987.

[10] Laura A. Sanchis. Test instance construction for NP-hard problems. Technical Report 206, University of Rochester, Computer Science Department, June 1987.

[11] Laura A. Sanchis. Maximum number of edges in connected graphs with a given domination number. *University of Rochester, Computer Science Department, internal document*, April 1988.

[12] Laura A. Sanchis. Test case construction for the dominating set problem. *University of Rochester, Computer Science Department, internal document*, revised August 1988.

[13] Laura A. Sanchis and Mark A. Fulk. Efficient language instance generation. Technical Report 235, University of Rochester, Computer Science Department, January 1988.

[14] P. Turan. An extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

[15] P. Turan. On the theory of graphs. *Colloquium Mathematicum*, 3:19–30, 1954.

RELATIVE COMPLEXITY OF EVALUATING AND CONSTRUCTING THE OPTIMUM OF MAXIMIZATION PROBLEMS

P. Crescenzi and R. Silvestri
Dipartimento di Matematica
Università di Roma "La Sapienza"
Piazzale Aldo Moro 5, 00185 Rome, Italy

ABSTRACT: We investigate how non constructive optimization functions, ϕ, i.e. functions that do not yeld an optimum solution, relate to the corresponding constructive ones, ϕ_C. We show that, for several maximization problems, ϕ_C is harder to compute than ϕ w.r.t. the obvious generalization of the many-one reduction. We also prove that there exists a problem such that ϕ is poly–time computable and ϕ_C is not iff the γ-reduction is different from the many-one reduction. Furthermore, we introduce special kinds of one-way functions whose existence is strictly related to the previous result. Finally, we relate these and other questions to the relative complexity of checking and evaluating general functions.

Keywords: optimization problems, many-one reducibility, NP-complete problems, Turing reducibility.

1. INTRODUCTION

Any optimization problem can be solved in two different ways: constructively and non constructively. For example, the problem of finding the maximum clique in a given graph could be solved either by "constructing" a maximum clique (i.e. listing the nodes of the clique) or by "evaluating" the size of a maximum clique. In the case of NP optimization problems (such as MAXCLIQUE, TSP, etc...) the cost function is polynomially computable; hence if we can construct an optimum solution in poly-time, then we can evaluate the maximum in poly-time too. What about the viceversa? If we know the cost of an optimum solution,generally it does not seem possible to construct such an optimum. And, even if we could compute in poly-time the cost of the optimum solutions of arbitrary instances of the problem, the problem of constructing effectively an optimum does not seem to be solvable in poly-time. In other words, as observed in [2] (where constructive functions are called non numeric), constructive solutions seem to be harder to compute, since they yield additional information.

In this paper, we study the relative complexity of evaluating and constructing the optimum of maximization problems whose corresponding decisional problems are in NP. We also relate this question to the existence of special kinds of one-way function. Furthermore, we show that there is a strong relationship between our questions and the questions posed by Valiant [6] about the relative complexity of checking and evaluating a function.

The paper is organized in the following way. In section 2 we give basic definitions. In section 3 we prove that, for several problems, constructing an optimum is harder than evaluating the optimum, w. r. t. the generalized version of the metric reduction introduced in [3]. In section 4 we prove that there exists a problem such that we can compute the optimal cost in poly–time but we can not construct an optimal solution in poly-time iff the γ-reduction is different from the many-one reduction [4]. The result is obtained by introducing a special kind of one-way function, which are strictly related to Valiant's classes defined in [6]. In section 5 we study some relationships between our questions and the classes introduced by Valiant.

2. BASIC DEFINITIONS

First we introduce some notation. We indicate with (N)TM the class of (non deterministic) Turing machines. Given a string y, n(y) represents the position of y in the lexicographic ordering of $\Sigma^*=\{0,1\}^*$. FP is the class of poly-time computable functions. The function $<,>:\Sigma^*\times\Sigma^*\rightarrow\Sigma^*$ is in FP and its inverses are in FP too. Given z in Σ^*, z_1 and z_2 are two strings in Σ^* such that $<z_1,z_2>=z$. For any function f, we abbreviate $f(<x,y>)$ by $f(x,y)$.

A natural way to define a maximization problem is to specify the instances, the solutions (possible and effective) and the cost function of the problem [1]. Thus we give the following:

Definition 1: a maximization problem Φ is a quadruple (I,S,π,c) such that:

1) I is a subset of Σ^*, called the set of the instances of Φ; I is recognizable in poly-time;
2) S is a polynomial NTM such that any computation of S must finish with an output in Σ^*; for any $x\in I$, the set S(x) of the outputs of S on input x is called the set of possible solutions of the instance x of Φ;
3) π is a polynomial predicate such that, for any $x\in I$, π is defined on S(x); the set $E(x)=\{y\in S(x)|\pi(y)=1\}$ is called the set of effective solutions of Φ;
4) c is a polynomial function with range $\mathbb{N}$, such that, for any $x\in I$ and for any $y\in S(x)$, $c(x,y)=0$ if $y\notin E(x)$.

Let C(x) be the set of the costs of all the possible solutions: $C(x)=\{c(x,z)|z\in S(x)\}$. We associate to any problem Φ two solving-functions:

(i) the multi-valued function ϕ_c such that:

$\phi_c(x)=\{y|c(x,y)$ is a maximum of $C(x)\}$;

(ii) the single-valued function ϕ such that $\phi(x)$ is equal to the maximum of $C(x)$.

Definition 2: MAX is the set of all maximization problems.

In our context it is useful to define maximization problems more concisely: a maximization problem is a couple $\Phi=(p,c)$ where p is a polynomial and $c: \Sigma^*\times\Sigma^*\rightarrow\mathbb{N}$ is a function computable in poly-time. Then $C(x)=\{c(x,y) \mid |y|\leq p(|x|)\}$. It is easy to verify that this characterization is equivalent to definiton 1. Thus, in the following pages, we will use both definitions.

Given a maximization problem Φ, the set $A_{\Phi}=\{<x,k> \mid \phi(x)\geq k\}$ is the language associated to Φ.

We first define three classes of multi-valued functions, introduced in [6].

Definition 3 : PEP_t is the class of the total multi-valued functions f such that , for any x, the lenght of the outputs is bounded by $p(|x|)$ and such that f can be "computed" by a polynomial TM, that is there exists a polynomial TM *T* such that, for any input x in the domain of f, $T(x)\in f(x)$.

Definition 4: PCP_t is the class of the total multi-valued functions f such that, for any x, the lenght of the outputs is bounded by $p(|x|)$ and such that the language:

$$L=\{(x,y) \mid y\in f(x)\}$$

is recognizable by a polynomial TM.

Definition 5 : $NPCP_t$ is the class of the total multi-valued functions f such that, for any x, the lenght of the outputs is bounded by $p(|x|)$ and such that the language:

$$L=\{(x,y) \mid y\in f(x)\}$$

is recognizable by a polynomial NTM. Thus, for any function f in $NPCP_t$, there exists a polynomial NTM *NT* such that:

i) for any x and for any $y\in f(x)$, there exists a computation of *NT* (x) yielding y; and

ii) for any x, each computation of *NT* (x) either outputs $y\in f(x)$ or does not terminate.

Valiant [6] has shown that if $NPCP_t= PEP_t$ then $P=NP\cap co\text{-}NP$. Furthermore, Long [4] has shown that $NPCP_t\neq PEP_t$ iff the gamma-reducibility ($\leq_\gamma$) is not equivalent to the poly-time many-one reducibility ($\leq_m$). We will use these two results later. Let XP_t be a class of multi-valued functions. We indicate with XP_{st} the subset of XP_t of single-valued functions.

The class of the functions ϕ associated with the maximization problems has been studied in [3]. In order to study the relative complexity of ϕ and ϕ_c we introduce a generalization of the metric reduction introduced in [3].

Definition 6 : let f and g be two multi-valued functions. A metric reduction from f to g is a pair of poly-time computable functions (T_1,T_2) such that: for all x and for any $y \in g(T_1(x))$, $T_2(x,y) \in f(x)$.

Given f and g, if there exists a metric reduction from f to g then we will write $f \leq g$. Metric reduction is reflexive and transitive. It is easy to verify that if $g \in PEP_t$ and $f \leq g$ then $f \in PEP_t$. It is also easy to verify that for any MAX-problem Φ it results that $\phi \leq \phi_c$. In the next section, we shall prove that, if $P \neq NP$, the viceversa does not hold in general.

3. CONSTRUCTING AND EVALUATING W.R.T. THE MANY-ONE REDUCTION

We first show that, given a MAX-problem Φ we can build a Φ' such that to construct an optimum of Φ is as difficult as to evaluate the optimal cost of Φ'.

Theorem 1: for any Φ in MAX, there exists a Φ' in MAX such that $\phi_c \leq \phi'$.

Proof: let $\Phi=(p,c)$ be any problem in MAX. Define a new maximization problem $\Phi'=(p',c')$ where $p'=p$ and $c'(x,y)=[n(1^{p(|x|)})+1]c(x,y)+n(y)$. Obviously, ϕ'_c is single-valued and $\phi'_c(x) \in \phi_c(x)$. Then, defining $T_1(x)=x$ and $T_2(x,y)=y$, we have that $\phi_c \leq \phi'_c$. Moreover, given x and z, we can compute in poly-time the y such that $c'(x,y)=z$. Then $\phi'_c \leq \phi'$. ■

The next results show us that going from Φ to Φ' is necessary for a large class of maximization problems.

Theorem 2: if $P \neq NP$, then there exists a Φ in MAX such that $\neg(\phi_c \leq \phi)$.

Proof: let us define, informally, the problem Φ: given a boolean formula x, $\phi(x)$ is 1 if x is satisfiable, it is 0 otherwise; and $\phi_c(x)$ is the set of assignments that satisfies x if x is satisfiable; otherwise $\phi_c(x)$ is the set of all possible assignments. Let us suppose that there exists a metric reduction from ϕ_c to ϕ, that is there exists a pair of poly-time computable functions (T_1,T_2) such that: $T_2(x,\phi(T_1(x))) \in \phi_c(x)$. We recall that

$\phi(T_1(x))$ can be either 0 or 1. Thus, if x is satisfiable then any assignment in $\phi_c(x)$ must statisfy x. Then either $T_2(x,0)$ or $T_2(x,1)$ must satisfy x. Viceversa, if x is not satisfiable then no assignment in $\phi_c(x)$ can satisfy x. Then neither $T_2(x,0)$ nor $T_2(x,1)$ can satisfy x. So the following simple algorithm:
compute $T_2(x,0)$; if it satisfies x then answer yes else compute $T_2(x,1)$; if it satisfies x then answer yes else answer no;
would solve SAT in poly-time. ■

The previous proof can be generalized to obtain the following result.

Theorem 3: Let Φ be an MAX-problem such that, for any x, $|\phi(x)| \in O[\log(|x|)]$. If $P \neq NP$ and A_Φ is NP-complete, then $\neg(\phi_c \leq \phi)$.

4. CONSTRUCTING AND EVALUATING W.R.T. THE TURING REDUCTION

In the last section, we saw that, w.r.t. the metric reduction, the constructive function can be computationally harder than the non constructive one.However, it is easy to verify that the proof of theorem 2 is no longer valid if we use the Turing reducibility ($\leq_T$) generalized in the same way in which we have generalized the metric reduction (showing, in theorem 2, that $\phi_c \leq_T \phi$ is very similar to showing that SAT is self-reducible [5]). Then, a natural question that arises is whether, for any maximization problem, the constructive function is Turing reducible to the non constructive one.

In this section, we want, under some reasonable assumptions, to construct a MAX-problem such that the optimal cost is poly-time computable while it is not possible to construct such an optimum in poly-time. The difficulty lies in the following corollary of theorem 1.

Corollary 1: let Φ be an MAX-problem. If A_Φ is NP-complete, then $\phi_c \leq_T \phi$.

In order to prove our results, we define special kinds of one-way functions.

Definition 7: a function f is called many-one-way (MOW) if:
(1) $f \in FP$;
(2) for any g, such that, for any y in the range of f, $f(g(y))=y$, $g \notin FP$;
(3) f is honest, that is there exists a polynomial p such that, for any x in the domain of f, $|x| \leq p(|f(x)|)$.

If a MOW function is one-to-one then we say that it is a one-way (OW) function.

It is easy to show the following fact.

Fact: there exists a MOW function iff $P \neq NP$.

We say that a function f is a P-range and many-one-way (PRAMOW) function if it is MOW and its range belongs to P. Analogously, a OW function is said to be a PROW function if its range belongs to P. We call PRAMOW (PROW) the class of PRAMOW (PROW) functions. In a certain sense, PRAMOW (PROW) functions are MOW (OW) functions such that the intractability of their invertion does not depend on their range. The following results give evidence that PRAMOW functions exist.

Theorem 4: there exists a PRAMOW function iff $NPCP_t \neq PEP_t$.
Proof: let f be a PRAMOW function. Define a function g such that, for any y, $g(y)=\{x \mid f(x)=y\}$ if y belongs to the range of f; $g(y)=\emptyset$ otherwise. Since f is P-range and it is honest then $g \in NPCP_t$. If g belonged to PEP_t, then there would exist a polynomial TM *T* such that, for any x, *T* $(x) \in g(x)$. The function $h(x)=$*T* (x), then would contradict the conditon (2) of definition 7 for f.
If $NPCP_t \neq PEP_t$, let f be a function in $NPCP_t - PEP_t$. Let *NT* be the polynomial NTM defined at the end of definiton 8. Define a function h such that $h(x,y)=1x$ if y is a computation of *NT* (x) that terminates; $h(x,y)=0<x,y>$ otherwise. It is easy to verify that h satisfies (1) and (3) of definition 7. Furthermore, since f is total, the range of h is in P. Finally, if there existed a function g such that, for any z, $f(g(z))=z$, then, for any x, we could compute, in poly-time, a $w \in f(x)$ in the following way: evaluate $g(1x)=(x,y)$ and set w equal to the output of the computation y. Then f would belong to PEP_t. ■

From the observations at the end of definition 5, the following result follows.

Corollary 2: if $P \neq NP \cap co\text{-}NP$ then there exists a PRAMOW function.

The following result relates the existence of PRAMOW functions to the question about the relationship between the constructive and non constructive function of a maximization problem.

Theorem 5: there exists a PRAMOW function iff there exists a problem Φ

in MAX such that: $\phi \in FP$ and $\phi_c \notin PEP_t$.

Proof: let f be a PRAMOW function. Then we define the problem $\Phi(p,c)$ where: p is the polynomial witnessing the honesty of f and $c(x,y)=x$ if $f(y)=x$, it is 0 otherwise.
From the definition, it follows that for any x, $\phi(x)=x$ if x belongs to the range of f, $\phi(x)=0$ otherwise. Since f is PRAMOW, then $\phi \in FP$. Let us suppose that $\phi_c \in PEP_t$. Let T be the polynomial machine such that for any x $T(x) \in \phi_c(x)$. From the definition of c, it follows that $f(T(x))=x$ for any x in the range of f. But this contradicts condition (2) of definition 7.
Viceversa, let $\Phi=(p,c)$ be an MAX-problem such that $\phi \in FP$ and $\phi_c \notin PEP_t$. We define the function f in the following way:
$f(x,y)=<x,1^{p(|x|)+1}>$ if $c(x,y)=\phi(x)$; $f(x,y)=<x,0y>$ otherwise. Since ϕ is in FP then f is in FP too. It is also easy to verify that f is honest. Furthermore, for any x and y, we can easily verify in poly-time if $<x,y> \in f(\Sigma^*)$. Finally, if there existed a function g in FP such that, for any z in the range of f, $f(g(z))=z$, then we could define $h(x)=(g(x,1^{p(|x|)+1}))_2$. Obviously, h is in FP and for any x $h(x) \in \phi_c(x)$. But this contradicts the hypothesis that $\phi_c \notin PEP_t$.■

Thus from theorems 4 and 5, and from the observations at the end of definition 5, we have the following result.

Corollary 3: $\leq_\gamma \neq \leq_m$ iff there exists a problem Φ in MAX such that: $\phi \in FP$ and $\phi_c \notin PEP_t$.

The previous corollary gives some evidence that, even with respect to the Turing reducibility, there exists a MAX-problem such that its constructive function is not reducible to the non constructive one, and so it is computationally harder in a very strong sense. But it also shows us that proving this statement is as hard as proving the inequivalence between gamma-reducibility and poly-time many-one reducibility.

5. ϕ_c AND ϕ W.R.T. THE VALIANT'S CLASSES

In this last section, we shall prove some results about membership of ϕ_c and ϕ in the classes of multi- and single-valued functions introduced in [6]. First of all, it is easy to prove the following result.

Theorem 6: Let Φ be a maximization problem. Then the following hold:
$\phi_c \in PEP_t \Rightarrow \phi \in PEP_{st} \Rightarrow \phi \in PCP_{st} \Rightarrow \phi_c \in PCP_t \Rightarrow \phi \in NPCP_{st} \Leftrightarrow$

$\phi_c \in NPCP_t$.

The next theorem shows that the converse of the previous implications are likely to be false, and, in any case, hard to prove.

Theorem 7:
(a) $\exists\Phi : \phi\in PEP_{st} \wedge \phi_c\notin PEP_t \Leftrightarrow PEP_t\neq NPCP_t \Leftrightarrow PRAMOW\neq\varnothing$;
(b) $\exists\Phi : \phi\in PCP_{st} \wedge \phi\notin PEP_{st} \Leftrightarrow PEP_{st}\neq PCP_{st} \Leftrightarrow PROW\neq\varnothing$;
(c) $\exists\Phi : \phi_c\in PCP_t \wedge \phi\notin PCP_{st} \Leftrightarrow PCP_{st}\neq NPCP_{st} \Leftrightarrow P\neq NP\cap co\text{-}NP$;
(d) $\exists\Phi : \phi\in NPCP_{st} \wedge \phi_c\notin PCP_t \Leftrightarrow \exists\Phi : \phi_c\in PCP_t \wedge \phi\notin PCP_{st}$.
Proof: (a) It follows from theorems 4 and 5.
(b) It is obvious that

$$\exists\Phi : \phi\in PCP_{st} \wedge \phi\notin PEP_{st} \Rightarrow PEP_{st}\neq PCP_{st}.$$

Let us suppose that $PEP_{st}\neq PCP_{st}$. Let f be a function in $PCP_{st}\text{-}PEP_{st}$. Define the function g in the following way: g(x,y)=1x if f(x)=y; g(x,y)=0<x,y> otherwise. Evidently, g is a PROW function. Finally, we have to show that

$$PROW\neq\varnothing \Rightarrow \exists\Phi : \phi\in PCP_{st} \wedge \phi\notin PEP_{st}.$$

Let f be a PROW function. Define the problem Φ=(p,c) where p is the polynomial witnessing the honesty of f and c(x,y)=y if f(y)=x, it is 0 otherwise. From the definition, it follows that for any x in the range of f, we have: $f(\phi(x))=x$. Since f is PROW, then $\phi\notin PEP_{st}$. And it is obvious that $\phi\in PCP_{st}$.
(c) From the fourth implication of theorem 7 it follows that

$$\exists\Phi : \phi_c\in PCP_t \wedge \phi\notin PCP_{st} \Rightarrow PCP_{st}\neq NPCP_{st}.$$

Let f be a function in $NPCP_{st}\text{-}PCP_{st}$. Let us consider the set $G_f=\{(x,y)|f(x)=y\}$. Obviously $G_f\in NP\text{-}P$. Let *NT* be the NTM that recognizes G_f. Then co-G_f is recognized in poly-time by the following nondeterministic algorithm ((x,y) is the input):
generate all the couples (x,z) with $z\leq p(|x|)$; *simulate NT(x,z), for any (x,z); if NT(x,z) rejects then reject; otherwise accept iff* $z\neq y$.
Thus $G_f\in NP\cap co\text{-}NP\text{-}P$.

Finally, we have to show that

$$P\neq NP\cap co\text{-}NP \Rightarrow \exists\Phi : \phi_c\in PCP_t \wedge \phi\notin PCP_{st}.$$

Let $A\in NP\cap co\text{-}NP\text{-}P$. It is easy to define a NTM *NT* such that, for any x, there exists a computation of *NT* (x) that computes the characteristic function of A and any computation of *NT* (x) either computes the characteristic function of A or does not terminate. Let us define the problem Φ=(p,c) where p is the polynomial bound on the lenght of the computations of *NT* and c(x,y) is the output of y, if y is a computation of

NT (x) that terminates; it is 0 otherwise. Evidently $\phi_c$is in PCP_t. Furthermore, ϕ cannot belong to PCP_{st} because otherwise A would belong to P.

(d) Let $\Phi=(p,c)$ be a MAX-problem such that $\phi\in NPCP_{st}$ and $\phi_c\notin PCP_t$. From theorem 7, it follows that $\phi_c\in NPCP_t\text{-}PCP_t$, that is the set $G_c=\{(x,y)|y\in\phi_c(x)\}$ belongs to NP-P. By similar arguments used in (c), we can easily show that G_c belongs to co–NP.Then $P\neq NP\cap co\text{-}NP$, and from (c) it follows that $\exists\Phi : \phi_c\in PCP_t \wedge \phi\notin PCP_{st}$. Viceversa, let $A\in NP\cap co\text{-}NP\text{-}P$. Then there exists a polynomial p and a polynomial predicate R such that:

$$x\in A\Leftrightarrow(\exists y)[|y|\le p(|x|)\wedge R(x,y)=1].$$

Let us define the problem $\Phi=(p,c)$ where $c(x,y)=R(x,y)$. From the definition it follows that, for any x:

$\phi_c(x)=\{y||y|\le p(|x|)\wedge(R(x,y)=1\vee(\forall z\le p(|x|))[R(x,z)=0])\}$. Let G_c be the set defined previously. Then

$$(x,y)\in G_c\Leftrightarrow|y|\le p(|x|)\wedge[R(x,y)=1\vee x\notin A].$$

Since $A\in$ co-NP then $G_c\in NP$. Furthermore, G_c cannot belong to P. In fact, if it was then A would belong to P too, since

$$x\in A\Leftrightarrow R(x,0)=1\vee[R(x,0)=0\wedge(x,0)\notin G_c].$$

Thus $\phi_c\in NPCP_t\text{-}PCP_t$. From (c) and from theorem 7 it follows that

$$\exists\Phi:\phi_c\in PCP_t\wedge\phi\notin PCP_{st}\Rightarrow\exists\Phi:\phi\in NPCP_{st}\wedge\phi_c\notin PCP_t.$$ ■

6. OPEN QUESTIONS

We propose the following directions for further work.

1) Find conditions for the existence of a MAX-problem Φ such that $\phi\notin PEP_{st}$ and $\neg(\phi_c\le_T\phi)$.

2) Look for a reduction (different from the metric one) stronger than the Turing reduction, such that there exists a MAX-problem whose associated language is NP-complete and whose constructive function is computationally harder than the non constructive one w.r.t. that reduction.

3) Define a notion of completeness for constructive functions such that it could characterize the approximability properties of optimization problems.

REFERENCES

[1] AUSIELLO G., MARCHETTI-SPACCAMELA A. and PROTASI, M. "Combinatorial problems over power sets", Calcolo, 16,4, 539-557, 1980

[2] GASARCH W. "The complexity of optimization functions", Tech. Report 1652, Dept. of Computer Science, University of Maryland, 1986

[3] KRENTEL, M.W. “The complexity of optimization problems”, Ph.D.Thesis, Technical Report, Dept. Computer Science, Cornell University, Ithaca, NY

[4] LONG, T.J. “On γ-reducibility versus polynomial time many-one reducibility”, Theoretical Computer Science 14, 91-101, 1981

[5] SCHOENING, U. “Complexity and structure”, Lect. Notes in Computer Science 211, Springer Verlag, 1985

[6] VALIANT, L.G. “Relative complexity of checking and evaluating”, Information Processing Letters, 5, 1, 20-23, 1976

TESTING OF COMS CIRCUITS AT FUNCTIONAL LEVEL

LIMING ZHANG, LANG TONG, AND RUEY-WEN LIU
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

The testing of CMOS circuits is important for the development of VLSI technology. However, the conventional "stuck at fault" assumption can not be used for the modeling of stuck-open faults for CMOS circuits. Recently, nonclassical fault modeling and testing for some special CMOS circuits at gate level have been considered [1] - [7] . This paper provides a unified approach for general combinatorial CMOS circuits at functional level. Any combinatorial CMOS circuit is first partitioned to complex logic unit (CLU) which is the smallest unit that insures the input reachability and output observability. It is then shown that the fundamental testing set of such unit can be found if and only if some conditions are satisfied. Finally, the test for the entire CMOS circuit can be generated from the fundamental testing set, for example by the extended D-algorithm. An example is used to illustrate this method.

I. INTRODUCTION

The conventional stuck-at model widely accepted at logic gate level is no longer sufficient for CMOS circuit because stuck-open faults in a CMOS circuit can not be modeled by stuck-at faults. The main difficulty in stuck open fault testing is the introduction of memory into the memoryless system. The testing for nonclassical faults, especially stuck open faults has been a major research problem recently and many approaches have been proposed for special CMOS circuit testing [1]-[7]. However, to our best knowledge, little effort has been made to formalize this practical problem in a systematic manner, especially from circuit theoretical point of view.

This paper provides a unified approach for general CMOS circuit at a functional level. Any large combinatorial CMOS circuit can first be decomposed to complex logic units (CLU) which is the smallest unit that insures the input reachability and output observability. According to the path structure of each CLU, the fundamental testing set can be found. Extended D-algorithm can then be used to find all possible tests if it exists. In section I, we first study the testability and test generating problem of the function unit decomposed from the general large CMOS circuits. In section II, we shall discuss the testing problems in the context of entire system. The extension of D-algorithm is also introduced. Finally, examples are given to illustrate our approach.

II. NETWORK DECOMPOSITION AND TESTING PROPERTIES

The fault testability primarily depends on the reachability by the testing inputs and the observability from the erroneous output. To find the suitable decomposition for large system, we like to decompose the large system to the extend that the reachability and the observability are insured. In CMOS system, a large classes of logic implementation have such property. Among those, one class of logic units that stands out from others is complex logic units. In practice, such kind of logic units usually serves as the fundamental building block for more complicated systems because it allows for relatively complicated logic functions to be realized using minimum number of transistors. This in turn reduces the overall chip area consumption, allowing for high density circuit integration.

In this section, we shall first introduce the graphic representation of CMOS circuit, define the complex logic units from circuit graph point of view which will give us deeper understanding of such unit.

An edge is a branch between two nodes. A path is a connected sequence of edges. The graph of a MOS transistor is one when its source-node is connected with its drain-node to become an edge, and its gate-node becomes an isolated node. A graph of a CMOS circuit is the one when all MOS transistors are replaced by their transistor graphs. An edge of a CMOS graph may now consist of one transistor-edge, or many transistor-edges. A CMOS circuit graph consists of four kinds of "exterior" nodes: the output-nodes, the ground node, or the N-node, the power supply node, or the P-node, and the input-nodes. A P-path is one which connects the P-node to the output-node, and an N-path is the one which connects the N-node to the output-node. A P-path is usually made of p-MOS transistors, and N-path is made of n-MOS transistors.

In this paper, we will consider the case of multiple-inputs and single output with no feedback.

Definition 1

A complex logic unit (CLU) is a CMOS circuit when all input-nodes are removed and every remaining edge is either an element of a P-path or an N-path, but not both.

A typical CLU graph is shown in Figure 1

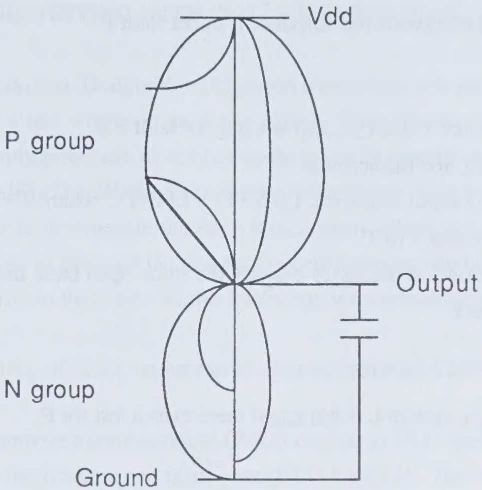


Fig1. A schematic diagram of CLU circuit.

The following input-output properties of fault free complex logic unit are assumed:

i) As a function of input, each transistor-edge in the CLU can be either open or short. An edge is conducting when it is short and nonconducting when it is open.

ii) For each input t, there is one (or more) P-path(s) or one (or more) N-path(s) conducting, but not both. If a P-path is conducting, we say the output is 1. Similarly, if an N-path is conducting, the output is 0.

iii) To avoid the trivial situation, we assume the output does not equal to 1 (or 0) for all inputs.

A CLU is said to have a single stuck-open fault if there is one and only one edge that is stuck-open, i.e., it is open for all input t. Note that if there exists an input t which causes all P-paths and N-paths open (under the stuck-open condition), then its output remains the same as the previous one. This is due to the memory property of the output capacitor. (see Fig. 1)

With the above definitions and input output properties of CLU, we are in the position to define the test and the testability of stuck-open fault, which is different from the stuck-at fault case.

Definition 2

A logic system L is said to have fault F if there exist a finite input sequence $(t_1,t_2,...,t_n)$ such that

$$L_F[(t_1,t_2,...,t_n)] \neq L[(t_1,t_2,...,t_n)]$$

where $L_F[(.)]$ represents the output of L under fault F.

Definition 3

An input sequence $T=(t_1,t_2,...,t_m)$ is a test for fault F if

i) $L_F(T) \neq L(T)$, and furthermore

ii) Let T' be any input sequence, $L_F(T'\bullet T) \neq L(T'\bullet T)$, where $T'\bullet T$ represents the operation that concatenates T to T'.

Remark: (ii) in definition 3 is necessary because the stuck-open fault, unlike the stuck-at fault, introduces memory.

Definition 4

A fault F in logic system L is testable if there exist a test for F.

Theorem

A single stuck-open fault F at edge e in CLU is testable if there exists a test t such that all paths to the output-node not containing e is open while there exists at least one path to output node containing e is short, and conversely if T is a test and an input t in T will satisfy the above condition.

Proof:

1) Assume there is one such t satisfies the condition given in the theorem. Without loss of generality, let L(t) =1. Since we have assumed, in the I/O specification of CLU, that there is another input t' such that L(t')=0, we claim that T=(t't) is a test for F in CLU. To verify this, we simply check the output $L_F(T)$ and L(T) by using the I/O properties defined above.

2) Let a stuck-open fault F at edge e in CLU be testable. Then there exists a test $T=(t_1,t_2,...,t_m)$ such that $L_F(T) \neq L(T)$. Let t be the one in T that $L_F(t) \neq L(t)$. Since there is only one stuck-open fault in CLU, then t must make at least one path containing e to output node conducting under fault free situation. Otherwise, we would have the contradiction $L_F(t) = L(t)$. Without loss of generality, let this path be a P-path. Then, under t, none of the N-paths are conducting. Suppose that there exists a P-path not containing e conducting under t, then the output will be determined by this path whether edge e is open or not. In other words, $L_F(t) = L(t)$, which draws contradiction. Therefore, none of the paths not containing e is conducting. ###

Remark:

The stuck-open fault in graphic notation of CLU actually represents a number of equivalent faults in the real circuit, such as a conducting transistor, lines between transistors, conducting transistors in series, etc.

III TESTING WITH CLU AS SUBFUNCTION

The reason that D-algorithm [8] played a important role in stuck at fault testing is that it can find a test whenever such test exists. Since the stuck open fault became a major issue, many proposals have been made to use D-algorithm for generating test for open fault. ([5],[6],[7].) While some methods involving changing the characteristic of the components to incorporate the open fault, many others are strictly at simple gate level. We propose to generate the test for the entire system from the tests obtained from subfunction, CLU in this case. In the following, we summarize our approach and give an example.

The testing of stuck -open faults of combinatorial CMOS circuits consists of three steps:

1. Decompose a combinatorial CMOS circuits to CLU components.
2. Find the fundamental testing set (FTS) for CLU. The fundamental testing set is defined as the set of all tests for the particular fault in CLU
3. Find a test for the entire CMOS circuit by using an extended D-algorithm.

Remark:

Obviously, the above approach is sufficient to find a test if such test exists. In [9], we will also show that these steps are necessary.

These three steps can be best illustrated by an example.

Consider the example given in Figure 2 (a). It is a CMOS circuit for an adder. The input lines are A, B, C and the output lines are Carry and Sum. They satisfy the formula as follows:

$$Sum = ABC + AB'C' + A'B'C + A'BC'$$

$$Carry = AB + AB'C + A'BC$$

Here A', B', C' are the complement of A, B, C. The Sum is the binary add of A, B and C.

Step I .Decompose the CMOS circuit into CLU components.

The resulting circuit is given in Figure 2 (b). Clearly, this circuit has formed four strongly connected parts as shown in Figure 2 (c). The decomposition process can always be achieved if circuit is unidirectional , i.e., no feedback.

Step II. Find the fundamental testing set of strongly connected components

A fundamental testing set is the complete set of testing signals whose length can not be further reduced [9]. Since a stuck- open fault of a CMOS circuit can be modeled by a sequential circuit with one shift - register [1], the length of the testing signal in the fundamental testing set is bounded by two [9]. We need only to find all the testing signals whose length is no longer than two.

Suppose that the stuck-open fault P of component (1) , is located as shown in Figure I (a) . Then it can be shown that (A,B,C) = (1,1,0) is the only input vector which satisfies the conditions of the theorem . Hence this open fault is testable.

Now, we want to find the fundamental testing set from the input

$$t = (A,B,C) = (1,\ 1,\ 0)$$

Note that under t, the output G of component (1) is zero in the fault free case. Now we want to find all possible input signals which will result an output which is opposite to that, i.e. G=1. In this example, we have the set

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Anyone of the above signal followed by t is a testing signal for the fault when P path is open, Hence the fundamental testing set is given by the following Table.

			fault free case	fault case
A	B	C	Output of part 1	Output of part 1 fault
(11)	(01)	(00)	(10)	permanent '1'
(01)	(11)	(00)	(10)	permanent '1'
(01)	(01)	(10)	(10)	permanent '1'
(01)	(01)	(00)	(10)	permanent '1'

Table I . The fundamental testing set for part 1 when path P is open

Step III. Find a test for the combinatorial CMOS circuit

An extended D- algorithm is used to find the test of entire CMOS circuit from the fundamental testing set . The extended D - algorithm contains three steps. They can simply be illustrated in the example. We assume that only the output Carry is used.

(1) Establish a primary D-cube[5] of fault P from Table I, where the fault can be observed at the output line during the second time interval. So the primary D- cube is

	A	B	C	G
t_c	1	1	0	D'

where D' represents an error happened at the output line G, which is '0' in normal case and '1' in fault case. D represents an error ,which is '1' in normal case and '0' in fault case. t_c is the primary D-cube.

(2) From the D-cube , find the D-path, i.e. the propagation of error D or D' from output G of part 1 to output Carry , which is given in Table II.

	A	B	C	G	output of Carry
t_c	1	1	0	D'	
				D'	D
	1	1	0	D'	D

Table II. D- path propagation

In this special case the input vector to the fault unit, i.e. part 1, is the same as the primary input vector . So the primary input vector is simply

$$(A, B, C) = (1, 1, 0)$$

In general, a singular cover [8] has to be used to find the primary input vector.

(3) According to the testing set of part 1, the first time interval is G='1'. So the input vector (A, B, C) is:

A	B	C	G
x	0	0	1
0	x	0	1
0	0	x	1

Here $x \varepsilon [0,1]$

Following the singular cover from the input of part 1 to the primary input the complete test for open path P is

$$I = (t_1, t_2) \text{ where } t_1 = (0,0,x), (x,0,0), (0,x,0), t_2 = (0,1,1)$$

IV CONCLUSION

In this paper, a unified approach is proposed to the problem of stuck-open fault testing in CMOS circuit. The testing problem are studied from circuit theoretical point of view. The theoretical results concerning testability are established.

REFERENCES

[1] Wadsack, R. L., "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits" The Bell Syst. Tech. J., May-June, 1978. pp14491473

[2] ELzig, Yacoub M, "Automatic Test Generation For Stuck-open Faults in CMOS VLSI" The 18th Design Automation Conference, Nashvile, June, 1981. IEEE 81 CH1643-6, pp 347-354.

[3] Y.M.Elziq and R.Cloutier, " Functional Level Test Generation for Stuck-open Faults in CMOS VLSI" in 1981 IEEE International test conf. pp. 536-546.

[4] Chandramouli, R., " On Testing Stuck-open Faults" 1983 FTCS Fault Tolerant Computing, pp258-265.

[5] Jain, Sunil K., "Test Generation For MOS Circuits Using D-algorithm" 20th Design Automation Conference, 1983. pp64-70[8] D. K. Pradhan , " Fault - Tolerant Computing Theory and Techniques" Prentice - Hall Englewood Cliffs, New Jersey, 1986.

[6] D. K. Pradhan , " Fault - Tolerant Computing Theory and Techniques" Prentice - Hall Englewood Cliffs, New Jersey, 1986.

[7] Sami A. Al-Arian and D.P. Agrawal, " Physical Failures and Fault Models of CMOS Circuits " IEEE trans. on Circuits and System Vol. CAS-34 No. 3, March, 1987. pp.269-279.

[8] J. Paul Roth ," Diagnosis of Automata Failures: a Calculus and a Method " IBM Journal, July, 1966. 278-290.

[9] Lang Tong, Liming Zhang and Ruey-wen Liu , "A Necessary and Sufficient Condition for Digital Testing of a Unified Functional Model " Submitted for publication.

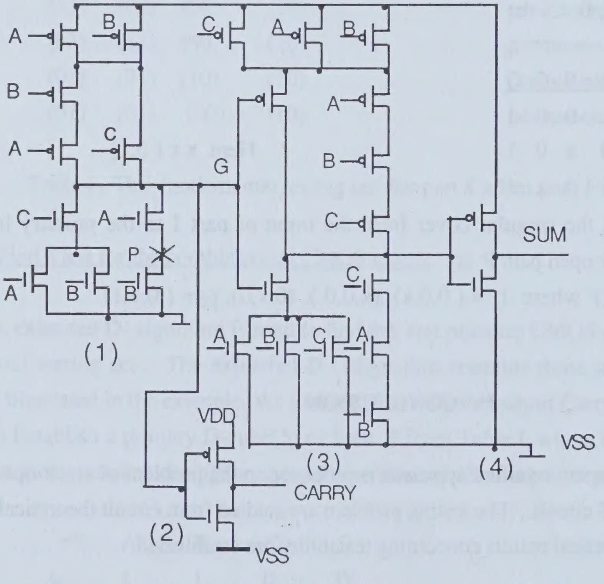


Figure 2 (a) CMOS network Adder

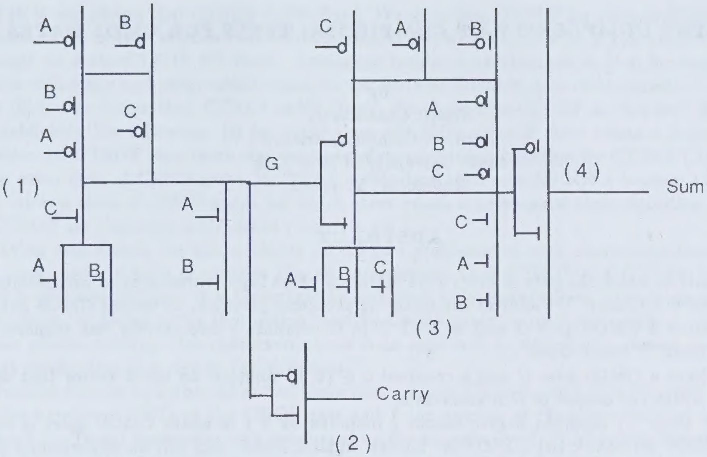

A
B
B
C
A
(1)
C
A
B
A
B
G
C
A
B
A
B
C
A
B
C
(4)
Sum
C
A
B
A
B
C
(3)
Carry
(2)

Figure I (b) CLU decomposition of Figure I (a)

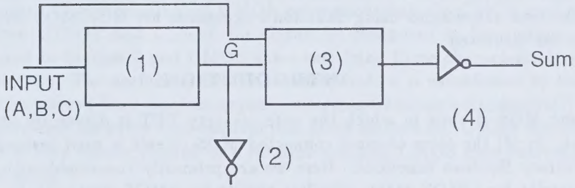

INPUT
(A,B,C)
(1)
G
(3)
Sum
(4)
(2)

Figure 2 (c) Decomposition diagram

ON THE COMPLEXITY OF COMPUTING TESTS FOR CMOS GATES

by
Sreejit Chakravarty [1]
Dept. of Computer Science
State University of New York
Buffalo, N. Y. 14260

ABSTRACT

CMOS circuits in which the gate of every FET is driven by an input variable or its complement are known as **CMOS Gates**. We address the following problems pertinent to testing CMOS gates.

CTSOF. Given a CMOS gate G and an FET T in G compute a two-pattern test sequence that detects the fault "T Stuck-Open".

CTSAF. Given a CMOS gate G and a constant $a \in \{0, 1\}$ compute an input vector that detects the fault in which the output of G is stuck-at a.

We show that: (i) bounded degree fanout (bounded by 2) in unate CMOS gates is enough to make CTSOF NP-Hard; (ii) CTSOF is "harder" than CTSAF; and (iii) an upperbound on the complexity of both CTSOF and CTSAF is $O(2^f \times m)$, where f is the number of fanout variables and m the number of FETs in the CMOS gate.

We also show that there exists a CMOS gate realization of Boolean functions named BC-CMOS circuits such that: (i) for every Boolean function there exists a BC-CMOS circuit; (ii) there exists a linear time algorithm for both CTSOF and CTSAF for BC-CMOS circuits; and (iii) CMOS gates derived from Boolean expressions using Shannon's expansion are BC-CMOS circuits. Implications of these results are discussed.

1. INTRODUCTION

MOS gates are MOS circuits in which the gate of every FET is driven by an input variable or its complement. In [8] the term channel connected MOS circuit is used instead. MOS gates can implement arbitrary Boolean functions. Here we are primarily concerned with CMOS gates and establish our results for CMOS gates. Similiar results for nMOS gates can be deduced from the discussion in this paper. Terms like NP-Hard etc. are defined in [5] and we assume that the reader is familiar with them.

A CMOS gate implementing the Boolean function f over the set of input variables $\{X_1, \ldots, X_n\}$ consists of a pullup network of PFETs and a pulldown network of NFETs. The pulldown network conducts when the input combination is such that $f(X_1, \ldots, X_n) = 0$ and the pullup network conducts when the input combination is such that $f(X_1, \ldots, X_n) = 1$. Figure 1.1(a) depicts a CMOS gate for the Boolean function f defined below.

$$f(X_1, X_2, X_3) = \overline{(X_1 \bigvee (X_2 \bigwedge X_1) \bigvee (X_3 \bigwedge \overline{X_2}))}.$$

As discussed in [9] a sequence of two input vectors is required to test FET stuck-open faults in CMOS gates. We address the following two problems pertinent to the testing of CMOS circuits.

CTSOF. Given a CMOS gate G and an FET T in G compute a two-pattern test sequence that detects the fault "T Stuck-Open".

CTSAF. Given a CMOS gate G and a constant $a \in \{0, 1\}$ compute an input vector that detects the fault in which the output of G is stuck-at a.

If $< t_1, t_2 >$ is a two-pattern test sequence for an FET stuck-open fault then t_1 is the **initializing vector** and t_2 is the **test vector.**

---

[1] Research supported by NSF Grant No. MIP-8802592.

In [8] it was shown that CTSOF is NP-Hard. We show that CTSOF for **simple CMOS gates** is NP-Hard. This in turn implies that bounded degree fanout (bounded by 2) in unate CMOS gates is enough to make CTSOF NP-Hard. Analogous results were obtained in [3,4] for test generation and controllability and observability analysis for stuck-at faults in gate level circuits.

In [8] it was shown that CTSAF is NP-Hard. We show that CTSOF is "harder" than CTSAF by establishing the following: (i) for every class of CMOS gates if there exists a polynomial time algorithm for CTSOF then there also exists a polynomial time algorithm for CTSAF (Lemma 1.3); (ii) for every class of CMOS gates, if CTSAF is NP-Hard then so is CTSOF (Lemma 1.3); and (iii) there exists a class of CMOS gates for which there exists a polynomial time algorithm for CTSAF but CTSOF for this class is NP-Hard (Lemma 2.6).

Having established the intractability of the two problems we seek characterizations of CMOS gates such that if a gate G satisfies this characterization then CTSOF and CTSAF for G can be solved in polynomial time. It is important to seek such a characterization because it might lead to the evolution of design styles that makes "testing CMOS gates" a tractable problem. Using a more familiar phrase, seeking such characterizations is an approach to "integrating design and test". We present results that is a step in this direction.

We show that an upperbound on the complexity of both CTSOF and CTSAF is $O(2^f \times m)$ where m is the number of FETs in the CMOS gate and f the number of "fanout variables" in the gate (Lemma 3.3). To our knowledge no upperbound on the complexity of CTSOF and CTSAF have been reported in the literature. Observe that in our upperbound if $f \leq Log(m)$ then the upperbound is polynomial in the size of the CMOS gate. This therefore provides a characterization of CMOS gates for which tests can be computed in polynomial time. This characterization is somewhat weak in that we are unable to show that for every Boolean function there exists a CMOS gate that satisfies the above criteria.

We provide another characterization of CMOS gates such that if a CMOS gate G satisfies this characterization then CTSOF and CTSAF for G can be computed in linear time. This characterization is discussed in Section 3 and CMOS gates satisfying these characterizations are referred to as BC-CMOS circuits. The usefulness of this characterization is established by showing that all complex CMOS gates derived from Boolean expressions using Shannon's Expansion[7] belong to this class. This also shows that for every Boolean function there exists a BC-CMOS circuit. The class of BC-CMOS circuits is very closely related to a testable complex CMOS gate realization of Boolean functions discussed in [2].

Our formulation of the problems of interest here as well as the proofs rely very heavily on the notion of labelled graphs which are used to represent MOS gates. This is discussed in Section 1.1.

1.1. LABELLED GRAPHS

Labelled graphs are multigraphs with one source node SO; one sink node SI; no self loops; and labelled edges where the labels are either the names of input variables or their complement.

Labelled graphs are a representation of Boolean functions as discussed below. A **path** in a labelled graph G is a sequence of edges $e_1, \ldots, e_k$ where each edge $e_i = (u_i, v_i)$ is traversed in one direction, say u_i to v_i. Let $< n_1, \ldots, n_{k+1} >$ be the corresponding sequence of nodes then: n_1 is the source node SO; n_{k+1} is the sink node SI; $n_i = n_j$ if and only if $i = j$; and $v_i = u_{i+1} = n_{i+1}$.

An **l-path** *corresponding to a path* $p =< e_1, \ldots, e_k >$ is a sequence of labels $< a_1, \ldots, a_k >$ where for all $1 \leq i \leq k$, a_i is the label on the edge e_i. The **label set** of an l-path $l =< a_0, \ldots, a_{k-1} >$ is the set of input literals $\{ b_0, \ldots, b_t \}$ such that for all $0 \leq i \leq t$ there exists atleast one j such that $0 \leq j \leq k-1$ and b_i equals a_j. A **label set** $\{ b_0, \ldots, b_t \}$ **is consistent** if and only if for all $i \neq j, b_i \neq \overline{b_j}$. An **l-path** l is said to be **consistent** if and only if its label set is consistent. A **path is consistent** if and only if the corresponding l-path is consistent. In Figure 1.2 the path $< SO, a, b, c, SI >$ is consistent whereas the path $< SO, a, e, f, SI >$ is inconsistent.

Let p be a path in a labelled graph G; $l =< a_0, \ldots, a_{k-1} >$ be the l-path corresponding to p; and $v =< c_1, \ldots, c_n >$ be an input vector. Then, the **path** p is said to be **activated by an input vector** v if and only if the following condition is satisfied: for all $0 \leq i \leq k-1$, either a_i is X_j for some $1 \leq j \leq n$ and c_j equals 1, or a_i is $\overline{X_j}$ for some $1 \leq j \leq n$ and c_j equals 0. Let G be a labelled graph and $f(G)$ the Boolean function represented by G. An input vector v satisfies $f(G)$ if and only if v activates atleast one path in G.

A CMOS gate G is represented by two labelled graphs G_u, G_d. The labelled graph G_d (G_u) represents the pulldown (pullup) network. They are defined as follows: for every node in the pulldown (pullup) network there exists a node in G_d (G_u); the node corresponding to the output (V_{dd}) is the source node SO; the node corresponding to the ground (output) is the sink node SI; there exist an edge E between nodes u , v of G_d (G_u) if and only if there exist a transistor T between the corresponding nodes in the pulldown (pullup) network; and E is labelled by the input literal (*complement* of the input literal) driving the gate of T. The labelled graphs G_u, G_d of the CMOS gate of Figure 1.1(a) is shown in Figure 1.1(b) and Figure 1.1(c) respectively.

Define the two problems LG-PATH and LG-TRAVERSE as follows.

LG-PATH. Given a labelled graph G determine an input vector t, if one such input vector exists, such that t activates atleast one path in G.

LG-TRAVERSE. Given: (i) a labelled graph G; and (ii) an edge e in G. Determine if their exists an input vector v such that: (i) v activates atleast one path in G; and (ii) all paths activated by v traverses the edge e.

Let T be a PFET (NFET) in a CMOS gate G implementing the Boolean function f and e the edge in G_u (G_d) corresponding to the transistor T. Then, the sequence of two input vectors $< t_1, t_2 >$ is a test sequence for the fault "T stuck-open" if and only if t_1 activates atleast one path in G_d (G_u); t_2 activates atleast one path in G_u (G_d); and all paths activated by t_2 traverses the edge e. The sequence of input vectors $< X_1 = 1, X_2 = 0, X_3 = 0 >, < X_1 = 0, X_2 = 0, X_3 = 0 >$ is a test sequence for the PFET e_1 of Figure 1.1(a). Similiarly, an input vector t detects the fault in which the output of G is stuck-at 0 (1) if and only if t activates a path in G_u (G_d). The input vector $< X_1 = 1, X_2 = 0, X_3 = 0 >$ is a test vector for the fault in which the output of the gate of Figure 1.1(a) is stuck-at 1. From the above discussion we have the following observations.

Observation 1.1 Let T be an NFET (PFET) in a CMOS gate G and let G_u, G_d and e be as defined above. Then, $< t_1, t_2 >$ is a test for T Stuck-Open if and only if

(a) t_1 is a solution of the instance of LG-PATH denoted by G_u (G_d); and

(b) if t_2 is a solution of the instance of LG-TRAVERSE denoted by G_d, e (G_u, e).

Observation 1.2 Let G, G_u, G_d be as defined above. An input vector t detects the fault in which the output of G is stuck-at 0 (1) if and only if t is a solution to the instance of LG-PATH denoted by G_u (G_d).

Lemma 1.3.

(a) For every class of CMOS circuit, a polynomial time algorithm for CTSOF implies a polynomial time algorithm for CTSAF.

(b) For every class of CMOS circuit, CTSAF is NP-Hard implies that CTSOF is NP-Hard.

Proof. Follows from Observation 1.1 and 1.2. □

2. COMPLEXITY OF COMPUTING TESTS

We show that CTSOF remains NP-Hard for a very restricted class of CMOS gates viz. simple **CMOS gates**. This class of CMOS gates implements only **unate Boolean functions** and every input variable drives no more than two FETs. The result in this section therefore show that the presence of bounded degree fanout in unate circuits is enough to make CTSOF NP-Hard. We also show that CTSOF is "harder" than CTSAF.

Let f and g be two Boolean formulae. We say that g **covers** f, denoted by $f \leq g$, if and only if for all input vectors v, $f(v) = 1$ implies $g(v) = 1$. If a Boolean formula f is such that for every input variable X_i the total number of times $X_i, \overline{X_i}$ appears in f is bounded by 1 then we say that f is a **fanout free Boolean formula**. If the Boolean formula f is such that no variable appears in its complemented form we say that f is a **monotone** Boolean formula. Observe that monotone

Boolean formulae consist of only the Boolean AND and OR operators. The **size of a Boolean formula** F is the number of Boolean operators in F.

Lemma 2.1 Let f and g be two monotone, fanout-free Boolean formulae. Then, the problem of determining if $f \leq g$ is *Co*NP-Complete.

Proof Refer to [5].

□

Let G be a labelled graphs labelled by the set of input literals $\{X_1, \ldots, X_n\}$ or their complements; for all $1 \leq i \leq n$, $E_i = \{e_1^i, \ldots, e_r^i\}$ be the set of edges in G labelled by X_i; and $F_i = \{f_1^i, \ldots, f_t^i\}$ be the set of edges of G labelled by $\overline{X_i}$. Then G is said to be a **fanoutfree labelled graph** if and only if for all $1 \leq i \leq n$, $r + t = 1$; a **monotone labelled graph** if and only if for all $1 \leq i \leq n$, F_i is empty; and a **unate labelled graph** if and only if for all $1 \leq i \leq n$, either E_i is empty or F_i is empty.

Lemma 2.2 Given a monotone, fanout-free Boolean formula f for a Boolean function F a monotone fanout-free labelled graph G for F can be computed in time $O(N)$ where N is the size of f.

Proof It can be shown by induction[10] that Algorithm BF-LG, stated below, computes a monotone, fanout-free labelled graph G for F from a monotone, fanout-free Boolean formulae for f.

Algorithm BF-LG

Step 1. Construct a parse tree for f[1].

Step 2. Synthesize attributes bottom up, starting from the leaves. Each node X in the parse tree is the root of a parse tree for an expression for a subfunction f_X of f. The node X has two children Y, Z and is labelled by a Boolean operator. For each node X synthesize an attribute X^1 where X^1 is a labelled graph for f_X.

The synthesis rule to be used at a node is determined by the operator at the node X. The rule of Figure 2.1(a) is to be used when the node X in the parse tree is a leaf node labelled by an input variable X_i. Else, the attribute X^1 is synthesized from Y^1, Z^1 which are the attributes of the children. The rules in Figures 2.1(b), 2.1(c) are to be used when the node X is a non-leaf node and the node is labelled by the operators AND, OR respectively.

□

A labelled graph G is said to be a **simple labelled graph** if and only if : (i) G is a unate; and (ii) for all input variables X_i, the number of edges labelled by X_i or $\overline{X_i}$ is bounded by 2.

Lemma 2.3 LG-TRAVERSE for simple labelled graph is *Co*NP-Complete.

Proof. We show that the covering problem for monotone, fanout-free Boolean formulae is embedded in LG-TRAVERSE for simple labelled graph. From this and Lemma 2.1 the result follows.

Let f and g be two monotone, fanout-free Boolean formulae over the set of input variables $X_1, \ldots, X_n$ for the Boolean functions F and G respectively. From f and g compute the monotone labelled graphs G_f and G_g for the Boolean functions F and G as discussed in the proof of Lemma 2.2. Let y be an input variable that is disctinct from $X_1, \ldots, X_n$. Construct the labelled graph G shown in Figure 2.2. The labelled graph G and the edge e consitutes an instance of LG-TRAVERSE for simple labelled graphs. An input vector v satisfies this instance of LG-TRAVERSE if and only if $F(v) = 1$ and $G(v) \neq 1$. In other words, the above instance of LG-TRAVERSE is satisfiable if and only if G is not a cover for F.

□

Let L_u, L_d be the two labelled graphs representing a CMOS gate G. Then G is said to be a **simple CMOS gate** if and only if both L_u, L_d are simple labelled graphs.

Theorem 2.4. CTSOF for simple CMOS gates is NP-Hard.

Proof. The result follows from Observation 1.1 and Lemma 2.3. □

From the definition of simple CMOS gates and Theorem 2.4 it follows that bounded degree fanout (bounded by 2) in unate CMOS gates is enough to make CTSOF NP-Hard.

Lemma 2.5 . There exists a linear time algorithm for CTSAF for simple CMOS gates.

Proof. Refer to [10]. □

Theorem 2.6. There exists a class of circuits for which there exists a linear time algorithm for CTSAF but CTSOFfor this class is NP-Hard.

Proof. From Theorem 2.3 and Lemma 2.5 it follows that simple CMOS gates is one such class of CMOS gates. □

From Lemma 1.3 and Theorem 2.6 it follows that CTSOF is harder than CTSAF.

3. AN UPPERBOUND FOR COMPUTING TESTS

An input variable X_i is said to be a **fanout variable** if and only if the number of NFETs or PFETs driven by X_i and its complement $\overline{X_i}$ is greater than 1. We present algorithms for CTSOF and CTSAF and show that the algorithms run in time $O(2^f \times m)$, where f is the number of fanout variables and m the number of FETs in the circuit. From Observations 1.1 and 1.2 it follows that in order to establish this result it is enough to show that there exist $O(2^f \times m)$ algorithms for LG-PATH and LG-TRAVERSE.

We modify the definition of a fanoutfree labelled graph. A labelled graph L is a **fanoutfree labelled graph** if the following conditions are satisfied: an edge could be labelled either by 1, or by an input variable or the complement of an input variable; and for all input variable X_i there exists atmost one edge that is labelled either by X_i or $\overline{X_i}$. We next present FFLG-PATH, FFLG-TRAVERSE the two algorithms respectively for LG-PATH, LG-TRAVERSE for fanoutfree labelled graphs.

FFLG-PATH

(* Let L be the labelled graph; SO the source node of L; and SI the sink node of L. *)

Step 1. Compute a path p from SO to SI and let p be the sequence of edges $< e_1, \ldots, e_k >$.

Step 2. Compute an input vector t that activates p as follows. Let for all j, e_j be labelled by Y_j. If Y_j is X_k then assign 1 to X_k else, if Y_j is $\overline{X_k}$ then assign 0 to X_k.

FFLG-TRAVERSE

(* Let L be the fanoutfree labelled graph; $e = (u, v)$ the edge in L; SO the source of L; and SI the sink of L. *)

Step 1 Determine if there exists a path $p =< e_1, \ldots, e_k >$ in L such that all the edges of p are labelled by 1 and p does not traverse the edge e, as discussed below. If there exists such a path p then there does not exist an input vector satisfying LG-TRAVERSE.
To determine if such a path exists from L eliminate the edge e and all the edges that are not labelled by 1 to get the new labelled graph L'. Using depth first search determine if there exists a path from SO to SI in L'.

Step 2 Using a modified depth first search of L determine a path $p_1 =< e_1, \ldots, e_t >$ from SO to either u or v. Without loss of generality assume that p_1 terminates at the node u. Compute a path p_2 in L from the node v to the node SI, where $p_2 =< f_1, \ldots, f_r >$.

Step 3 Let p be the path $< e_1, \ldots, e_t, e, f_1, \ldots, f_r >$. Return the input vector t where each input variable X_k is assigned a constant as follows. Let X_k ($\overline{X_k}$) label the edge F in L. Then, if p traverses the edge F then X_k is assigned the value 1 (0) else X_k is assigned the value 0 (1).

We have the following Lemmas. The proof of Lemmas 3.1 and 3.2 is left to the reader.

Lemma 3.1 Algorithm FFLG-PATH is an $O(m)$ algorithm for LG-PATH for fanoutfree labelled graphs.

Lemma 3.2 Algorithm FFLG-TRAVERSE is an $O(m)$ algorithm for LG-TRAVERSE for fanout-free labelled graph.

Theorem 3.3. There exists an $O(2^f \times m)$ algorithm for LG-PATH and LG-TRAVERSE where f is the number of fanout variables and m is the number of edges in the labelled graph.

Proof Let $\{X_1, \dots X_n\}$ be the set of fanout variables of the given labelled graph L. Define an assignment to be an f-tuple $< c_1, \dots, c_f >$ where: for all $1 \leq i \leq f, c_i \in \{0, 1\}$.

For each assignment derive the fanout free labelled graph L' from L as follows. For every fanout variable X_i let $\{e_1^i, \dots, e_k^i\}$ be the set of edges labelled by X_i and $\{f_1^i, \dots, f_t^i\}$ be the set of edges labelled by $\overline{X_i}$. For all $1 \leq j \leq k$, if c_i is 1 then e_j^i is labelled by 1 else e_j^i is labelled by 0. For all $1 \leq j \leq t$, if c_i is 1 then e_j^i is labelled by 0 else e_j^i is labelled by 1. From the labelled graph so obtained eliminate the edges labelled by 0 to derive the graph L'.

To solve LG-PATH (LG-TRAVERSE) in time $O(2^f \times m)$, for each of the 2^f assignments derive the graph fanout free labelled graph L' from L as stated above. If for any of the assignments $< c_1, \dots, c_n >$ FFLG-PATH (FFLG-TRAVERSE) succeeds in finding a solution it returns an assignments of constants from $\{0, 1, X\}$ to the input variables that do not fanout. Here X denotes a don't care condition and an input variable is assumed to have the value X if it is not assigned a value by FFLG-PATH (FFLG-TRAVERSE). An input vector is derived from the vector returned by FFLG-PATH (FFLG-TRAVERSE) by assigning c_i to X_i, for all $1 \leq i \leq f$. The result follows from Lemma 3.1 (Lemma 3.2). □

4. BC-CMOS Circuits

In this section we present a class of CMOS gates named BC-CMOS circuits for which tests can be computed in linear time. We also show that this class of CMOS gates contain all the CMOS gates derived from Boolean expressions using Shannons expansion and is therefore a class of CMOS gates that occur frequently in practice. This also shows that there exists a BC-CMOS circuit for every Boolean function.

A CMOS gate G is said to be a **BC-CMOS** circuit if and only if the labelled graphs representing G are free, binary labelled graphs. **Free, Binary labelled graphs** are defined as follows. Let $X_1, \dots, X_n$ be the set of input variables.

1. If the labelled graph G consists of only the source and sink nodes then G is a binary labelled graph if and only if it is in one of the forms shown in Figure 4.1, for some X_i.

2. If the labelled graph G has atleast one node that is not the source node or the sink node then G is a binary labelled graph if and only if G can be constructed from $G_1, G_2, X_i, \overline{X_i}$ using one of the construction rules of Figure 4.2. In Figure 4.2 X_i is an input variable; G_1, G_2 are free, binary labelled graphs; and none of the edges in G_1, G_2 are labelled by either $X_i, \overline{X_i}$.

   In Figure 4.2(a) a new source node SO along with two edges (SO, u) and (SO, v) is added to G_1. Here, u and v are nodes in G_1 which are not necessarily distinct and one of u, v is the source node of G_1. The sink node of G_1 is the sink node of G. In Figure 4.2(c) - (f) u is the source node of G_1, G_2.

The labelled graph of Figure 4.3(b) is a binary labelled graph whereas the labelled graph of Figure 4.3(a) is not a binary labelled graph. We have the following lemma.

Lemma 4.1. Let G be a free, binary labelled graph; and p, q be any two distinct paths in G. For all input vectors v, if p activates the path p then v does not activate the path q.

Proof. Refer to [10].

□

Cor. 4.2. Every input vector activates atmost one path in a free, binary labelled graph.

Proof. Readily follows from Lemma 4.1.

□

Lemma 4.3. There exists a linear time algorithm for LG-TRAVERSE for free binary labelled graphs.

Proof. Let G be the free, binary labelled graph and e and edge in G. We construct in linear time a consistent path $p =< e_1, \ldots, e_t >$ in G that traverses the edge e. An input vector v that activates p is constructed in a manner that is similiar to Step 2 of FFLG-PATH. From Cor. 4.2 it follows that p is the only path activated by v. Therefore, v is a solution for the instance of LG-TRAVERSE. We have the following recursive procedure to construct a path p.

If G consists of only the source and the sink node then G must be of the form shown in Figure 4.1 which has only one path. Therefore this is the required path.

Else, G consists of more than two nodes. We have six different cases corresponding to the six construction rules of Figure 4.2. We discuss the algorithm for the construction rule of Figure 4.2(a). The algorithm for the other cases is similiar and left to the reader.

Case 1. The edge e corresponds to (SO, N) where N is a node in G_1. Let $\{m_1, \ldots, m_p\}$ be the set of edges, other than the edge e, that are incident on the node N. For any one of m_i, recursively construct a consistent path p' in G_1 that traverses m_i. Let p' traverse the sequence of edges $< f_1, \ldots, f_j, f_{j+1}, \ldots, f_r >$ where: $f_j = (N, M)$; and p' traverse the edge f_j from N to M. Then, return the path $p = < e, f_j, \ldots, f_r >$.

Case 2. The edge e is an edge in G_1 where u is the source node of G_1. Recursively, construct a consistent path $< f_1, \ldots, f_r >$ in G_1 that traverses the edge e. Then, return the path $p =< f_0, f_1, \ldots, f_r >$, where f_0 is the edge (SO, u).

□

Lemma 4.4 There exists a linear time algorithm for LG-PATH for free, binary labelled graphs.

Proof. Very Similiar to the proof of Lemma 4.3. □

Theorem 4.5. There exists a linear time algorithm for CTSOF for BC-CMOS circuits.

Proof. Let N_1 be the number of PFETs and N_2 the number of NFETs. Let T be the transistor that is stuck-open. Without loss of generality assume that T is an NFET and e is the edge in G_d corresponding to the FET T. Since G_u is a free, binary labelled graph from Cor. 4.4 it follows that LG-PATH for G_u can be solved in time $O(N_1)$. From Lemma 4.3 it follows that LG-TRAVERSE for G_d can be solved in time $O(N_2)$. From these and Observation 1.1 the theorem follows.

□

We next show that CMOS gates derived using Shannon's expansion are BC-CMOS circuits. Let f be a Boolean expression over the set of input variables $\{X_1, \ldots, X_n\}$. Then, the Shannon's expansion of f with respect to X_i is defined as follows.

$$\begin{aligned} f(X_1, \ldots, X_n) \;&=\; X_i \bigwedge g_1(X_1, \ldots, X_{i-1}, X_{i+1} \ldots X_n) \\ &\bigvee \overline{X_i} \bigwedge g_2(X_1, \ldots, X_{i-1}, X_{i+1} \ldots X_n), \end{aligned}$$

where g_1 is the projection of f where $X_1 = 1$ and g_2 is the projection of f where $X_1 = 0$. This gives an algorithm named CONVERT for computing a labelled graph L from a given Boolean Formula F.

Algorithm CONVERT

Step 1. If F is a variable X_i or the complement of a variable X_i then L is as shown in Figures 4.1(a), 4.1(b) respectively.

Step 2. Else, choose an input variable X_i with respect to which the expansion is to be carried out. From F derive the Boolean formulae G, H by substituting X_i respectively by 1, 0 and then eliminating the constants. Compute the labelled graph G_1 (G_2) from the expression G (H) if the expression is not a constant. Then L is as shown in : Figure 4.2(b) if G, H are not constants; Figure 4.2(c) (Figure 4.2(d)) if H (G) reduces to the constant 0; and Figure 4.2(e) (Figure 4.2(f)) if G (H) reduces to the constant 1.

Note that the class of free, binary labelled graphs contain labelled graphs that cannot be derived using Shannon's expansion. The reason for this is that CONVERT does not use Figure 4.2(a). The labelled graph of Figure 4.3(b) is an example of a binary, free labelled graph that cannot be derived by CONVERT from any Boolean expression.

REFERENCES

1. A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools.* Addison Wesley, 1986.
2. S. Chakravarty, "A Testable complex CMOS gate realization of Boolean Functions", Tech. Rep. No. 88-10, Dept. of Comp. Sc., SUNY @ Buffalo, N.Y.
3. H. Fujiwara, "Computational Complexity of Controllability/ Observability Problems for Combinational Circuits", *To appear in the 18th Fault Tolerant Computing Symposium*, 1988.
4. H. Fujiwara and S. Toida, "The complexity of fault detection problem in combinational circuits", *IEEE Trans. on Comput.*, Vol. C-31, No. 6, 1982, pp. 555-560.
5. M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the Theory of NPCompleteness* , W. H. Freeman and Co., 1979.
6. H. B. Hunt III, and R. E. Stearns, "Monotone Boolean formula, Distributive Lattices, and the Complexity of Logics, algebraic structures, and computation structures", in *STACS86, Third Annual Symposium on Theoritical Aspects of Computer Science*, Vol. 210, *Lecture Notes in Computer Science*, Springer Verlag,1986. Eds. B. Monien, and G. Vidal-Naquet, pp. 277-291.
7. A. Mukhopadhyaya, "Introduction to nMOS and CMOS VLSI Systems Design". Prentice Hall, 1986.
8. F. Najm and I. Hajj, "The Complexity of Test Generation at the Transistor level", Report No. UILU-ENG-87-2280, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.
9. R. L. Wadsack, "Fault Modelling and Logic Simulation of CMOS and nMOS integrated circuits", *The Bell Systems Technical Journal*, Vol. 57, No. 5, pp. 1449 - 1474, May - June 1978.
10. S. Chakravarty, "On the complexity of computing tests for complex CMOS gates", Tech. Rep. No. 88-20, Dept. of Computer Science, State University of New York at Buffalo, NY 14260.

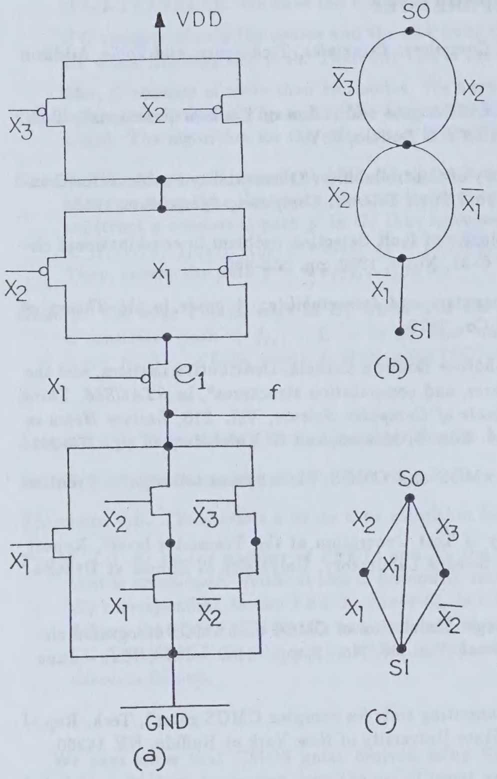

VDD
X3
X2
X2
X1
X1
e1
f
X1
X2
X3
X1
X2
GND
(a)
SO
X3
X2
X2
X1
X1
SI
(b)
SO
X2
X3
X1
X1
X2
SI
(c)

FIGURE 1.1

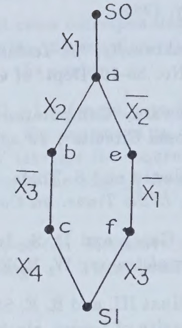

SO
X1
a
X2
X2
b
e
X3
X1
c
f
X4
X3
SI

FIGURE 1.2

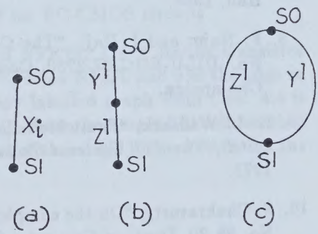

SO
Xi
SI
(a)
SO
Y1
Z1
SI
(b)
SO
Z1
Y1
SI
(c)

FIGURE 2.1

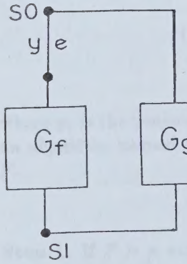

SO
y e
Gf
Gg
SI

FIGURE 2.2

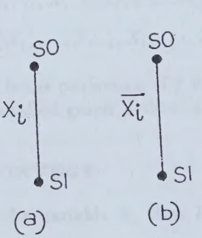
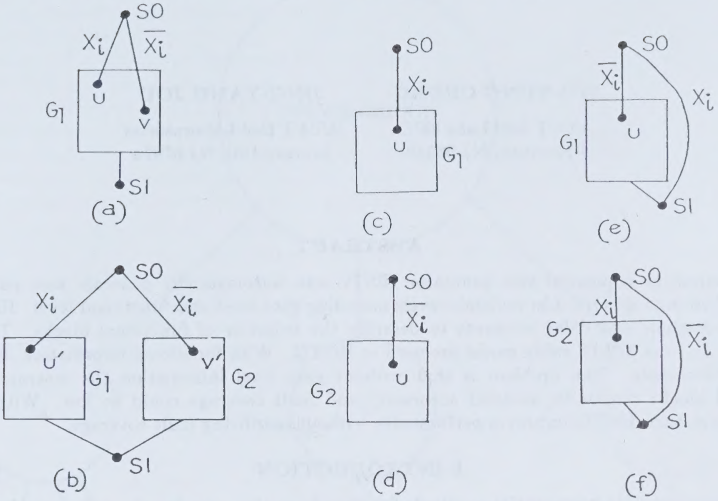
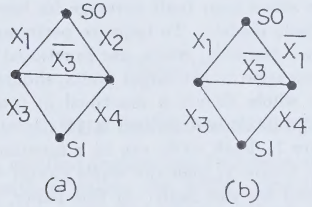

SO
Xi
SI
(a)
SO
Xi
SI
(b)

FIGURE 4.1



SO
X_i
$\overline{X_i}$
u
v
G_1
S1
(a)
SO
X_i
u
G_1
(c)
SO
$\overline{X_i}$
X_i
u
G_1
S1
(e)
SO
X_i
$\overline{X_i}$
u'
v'
G_1
G_2
S1
(b)
SO
X_i
u
G_2
(d)
SO
X_i
G_2
u
$\overline{X_i}$
S1
(f)

FIGURE 4.2



SO
X_1
X_2
$\overline{X_3}$
X_3
X_4
S1
(a)
SO
X_1
$\overline{X_1}$
$\overline{X_3}$
X_3
X_4
S1
(b)

FIGURE 4.3

A HIERARCHICAL SEQUENTIAL TEST GENERATOR

WU-TUNG CHENG
AT&T Bell Labs ERC
Princeton, NJ 08540

JING-YANG JOU
AT&T Bell Laboratories
Murray Hill, NJ 07974

ABSTRACT

A hierarchical sequential test generator HiSTG can automatically generate test patterns for sequential circuits described in multiple levels including gate level and functional level. HiSTG uses Boolean equations and delay elements to describe the behavior of functional blocks. The BACK algorithm and the SPLIT value model are used in HiSTG. With functional information, HiSTG has better performance. The problem is that without gate level information the internal faults of functional blocks cannot be modeled accurately and fault coverage could be low. With multiple levels information, HiSTG improves performance without sacrificing fault coverage.

I. INTRODUCTION

For many years, test generation methods for digital circuits were based on the gate level circuit description [1-7]. As VLSI circuits get larger and more complicated, these methods are becoming prohibitively expensive in terms of computation time. To combat these complexity problems, several functional-level test generation methods have been proposed in [8-13]. They are successful only for PLAs, memories and microprocessors by taking advantage of their special structures [8,9]. For general circuits [10-13], they are not practical because that the quality of tests generated based only on functional level information shows poor fault coverage for lower level faults due to the lack of comprehensive functional-level fault models. To improve performance without sacrificing fault coverage, hierarchical test generation methods, which use functional level information to improve performance and use gate level information to get target faults, should be used. During the process of hierarchical test generation, the whole circuit is described in functional level except that the faulty block which contains the target fault is described in gate level. For example, a hierarchical circuit description is shown in Figure 1. Each circle can be a functional description or a gate level description. If the target fault is at Circle h, then the whole circuit will consist of Circles a, h, i, and c when trying to generate a test for this fault. In this paper, a hierarchical sequential test generator HiSTG will be described. HiSTG uses single line stuck-at fault model.

Test pattern generation is trying to find test inputs to detect target faults. It is a search problem to find inputs which can satisfy some specific requirements. Therefore, during test generation process, it is very important to know what input combinations of one element can satisfy its output requirement. However, a functional block is usually described in forward manner. The only general functional description which can easily derive their reverse functions are Boolean equations. Although not all the designers use Boolean equations to describe the behavior of their circuits, Boolean equations are usually used as intermediate description within synthesis system to translate behavior description down to circuit layout. Thus, Boolean equations are usually available in design process using high-level synthesis tools. Therefore, HiSTG uses Boolean equations and delay elements to describe the behavior of all functional blocks. Previous work on test generation for Boolean equations was described in [14,15]. The work in [14] was based on the D-algorithm [1] and the work in [15] was based on the PODEM algorithm [5]. They both have problem at how to propagate sensitized values through functional blocks. Furthermore, they can handle combinational circuits only. HiSTG is based on the BACK algorithm [7] to eliminate fault propagation problem and to handle sequential circuits. We will briefly describe the BACK algorithm in Section II. How to take advantage of Boolean equations to speed up test generation process will be discussed in Section III. Experimental results of HiSTG will be given in Section IV. Concluding remarks will be presented in Section V.

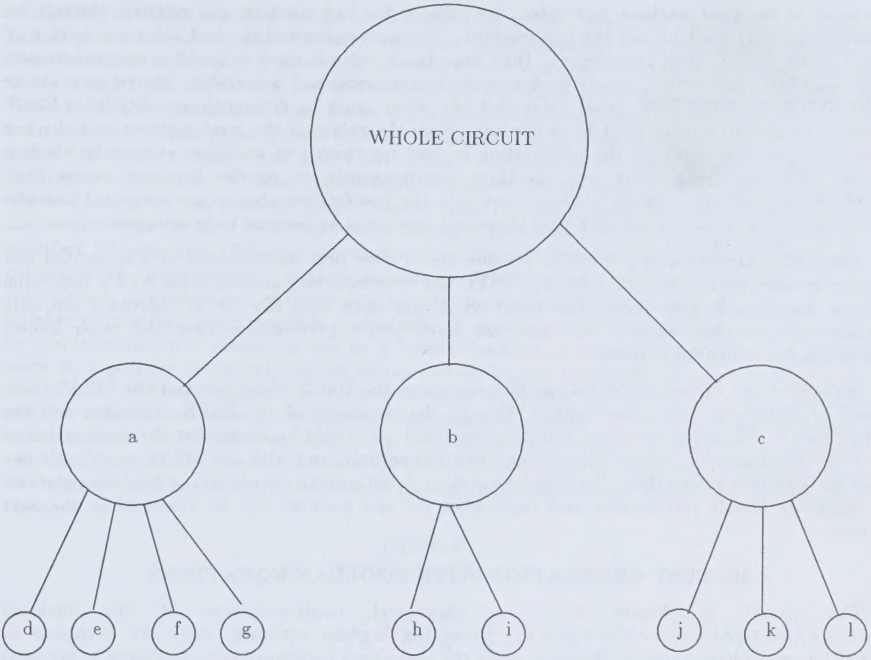


Figure 1.

II. THE BACK ALGORITHM

The BACK algorithm is a modified D-algorithm for sequential circuits. Instead of forward fault propagation to generate sensitized paths, the BACK algorithm generates sensitized paths by justifying a sensitized value at the preselected primary output. In other words, sensitized paths are generated from the sensitized primary output backward to the fault site. Without the forward propagation of the D-algorithm, the BACK algorithm only has backward line value justification through the circuit space and through the time domain. The preselection of a primary output to have a sensitized value is based on a new kind of testability measurement called *drivability*. The drivability of one line is measured by the difficulty of driving fault effect from the fault site to this line.

The procedure of the BACK algorithm can be summarized as follows.

1. Calculate the good machine controllability measurement.
2. Pick a target fault and calculate the bad machine controllability measurement and the drivability measurement.
3. Based on the drivability measurement, select a primary output to have a sensitized value (D or $\overline{D}$). Set the current time frame as $T=0$.
4. Justify all the values required at the current time frame.
5. If the state requirement matches the initial state, stop. Otherwise, set $T=T-1$ and go to Step 4.

To efficiently justify line values, the SPLIT value model [16] is used. The SPLIT model is an enhanced 9-valued model [3]. In the SPLIT model, each line has three values. They are *good value*:

the value at the good machine, *bad value*: the value at the bad machine, and *relation*: the relation between the good machine and the bad machine. The good value and the bad value can be 0, 1, X (don't care), and Z (high impedance). High impedance value is used to avoid generating vectors that cause bus conflicts and is only used around tri-state gates and bus nodes. The relation can be UNKNOWN, DIFFERENCE (good value and bad value must be different), and EQUIVALENCE (good value and bad value must be the same). With the values of the good machine and the bad machine separately recorded, the justification is done for these two machines separately which is simpler than justifying these two machines simultaneously as in the 9-valued model [3,4]. Furthermore, with the relation recorded, not only the justification choices are restricted but also more implication is carried out such that the search space can be reduced to be minimum.

The BACK algorithm and the SPLIT value model were first implemented in a sequential test generator called NSTG within AT&T [7]. NSTG has been used very successful for AT&T sequential circuits described in gate level. The result in [7] did show that the BACK algorithm not only requires less run-time memory but also has much better performance than the other known algorithms for sequential circuits.

Because of the success of NSTG, we like to extend the BACK algorithm and the SPLIT value model to high-level circuit descriptions. Due to the simplicity of the BACK algorithm and the SPLIT model, the extension can be easily implemented. It should be noted that the main activities of the BACK algorithm are justification and implication only, and with the SPLIT model only one machine is treated at one time. The implementation detail to take advantages of Boolean equations to efficiently handle justification and implication for one machine will be discussed in the next section.

III. TEST GENERATION WITH BOOLEAN EQUATIONS

The circuit in Figure 2 is a three-level implementation of the function $m=(a+\bar{b})(\bar{a}+b)(a+\bar{c})(\bar{b}+c)(\bar{c}+d)(c+\bar{d})$. From the Boolean equation, there are 6 choices to justify line m to have value 0. However, using the three-level implementation in Figure 2, the total number of choices will be $4*2=8$. It is obvious that to justify any line to have value 0 a minimum product-of-sum equation can give minimum number of choices. Similarly, in order to justify a line to have value 1, a minimum sum-of-product equation of that line should be used. It is worth noting that for large circuits with deep cascaded functional blocks, the reduction of the number of choices in justification will be huge. Although maintaining two kinds of equations has overhead, the improvements in justification and implication, which will be discussed later, are very significant.

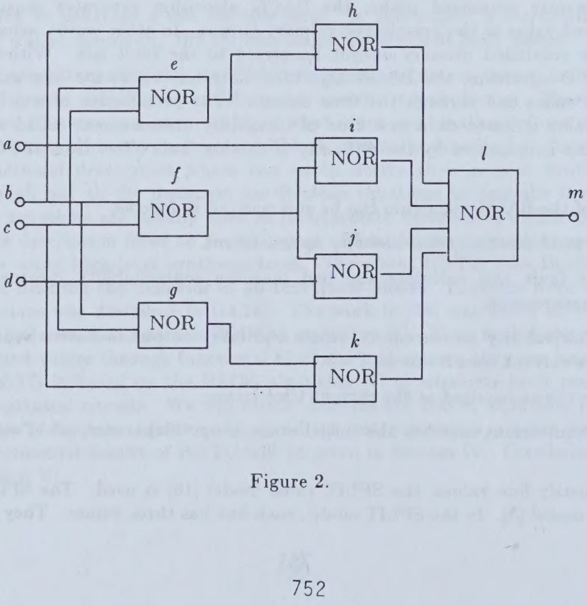


Figure 2.

In HiSTG, tables are used to represent Boolean equations. The table equivalent to a minimum sum-of-product form is called *ON-set*; while the table equivalent to a minimum product-of-sum form is called *OFF-set*. A minimum sum-of-product equation of Figure 2 is $m = abcd + \bar{a}\bar{b}\bar{c}\bar{d}$. Tables I and II are the ON-set and OFF-set of Figure 2. Each row is a choice to justify its output. However, the arrangement of these tables is not efficient for implication. For example, if line a has been assigned to logic value 1, some forward implication can be done in Figure 2 but not in Tables I and II. In Figure 2, node e and node i have value 0 determined, such that the search space for later justification is smaller. Without the capability to record the information that the first rows of both Tables I and II are not matched already cannot make the search space smaller for later justification and wastes time for later redundant table look-up. To record the match situation of each choice, modified ON-sets and OFF-sets are used in HiSTG. As shown in Tables III and IV, one extra column which contains *match symbols* and one *availability number* is added. This column is called *information column* and is changed dynamically during test generation process. The availability number records how many available choices left. A match symbol records the matching situation of its corresponding row choice. It can be a positive number, a negative number, 0 or not-matched mark &; a positive (negative) number means this choice can be matched (cannot be matched) and the number of inputs need to be further assigned to match this choice, 0 means the choice is matched already and & means this choice is not matched already. Negative number are assigned at two situations. One is when one choice was tried and failed; the other is when the equation output value is 1 (0), then the match symbols of OFF-set (ON-set) must be either negative numbers or &'s. The purpose of using negative number is for backward implication. That will be clear in later explanation.

Table I.
The ON-set of Figure 2.

a	b	c	d
0	0	0	0
1	1	1	1

Table II.
The OFF-set of Figure 2.

a	b	c	d
0	1	x	x
1	0	x	x
0	x	1	x
x	1	0	x
x	x	0	1
x	x	1	0

The operation of the information column can be explained by the following example. Tables III and IV are the initial situation. Table III shows that row 1 and row 2 both need 4 inputs assigned to be matched and there are 2 available choices. Right after input a was assigned to logic value 1, the information column is changed by checking the column under input a. There are 3 possible situations:

1. If one entry has don't care value x, nothing is changed.
2. If one entry has opposite value 0, & is marked at the corresponding match symbol and the availability number is decreased by 1. After that, if the new availability number is 1 and the output value is required, then the only available choice should be assigned immediately. This is a kind of backward implication. If the new availability number is 0, then the output can be assigned the opposite value. This operation solves the consensus problem of the other table [17].

3. If one entry has matched value 1 and if its corresponding match symbol is not &, then the absolute value of this match symbol is decreased by 1. After that, if the new number of this match symbol is 0 which means this choice is matched, the output value can be assigned. If the new number is -1, the unassigned input of this choice should be assigned a opposite value to make this choice not matched, otherwise conflict situation will happen. This is a kind of backward implication. This explains why including negative numbers in match symbols.

Tables V and VI are the situation after input *a* has logic value 1. Table V shows that row 1 is not matched already, row 2 still needs 3 inputs assigned to be matched, and there is 1 choice available. After that, if the output requests logic value 1, backward implication can be done by assigning inputs *b*, *c* and *d* to have logic value 1. For justification, an available choice can be selected by looking up information column only.

Table III.
The modified ON-set of Figure 2

a	b	c	d	
0	0	0	0	4
1	1	1	1	4
				2

Table IV.
The modified OFF-set of Figure 2

a	b	c	d	
0	1	x	x	2
1	0	x	x	2
0	x	1	x	2
x	1	0	x	2
x	x	0	1	2
x	x	1	0	2
				6

Table V.
The modified ON-set of Figure 2 after a=1.

a	b	c	d	
0	0	0	0	&
1	1	1	1	3
				1

Including negative numbers in match symbols to do backward implication has another important advantage. For example, Tables VII and VIII are the modified ON-set and OFF-set of one design after input $c=1$. If the output requests to be logic value 1, there are 5 choices available in the ON-set. With careful observation, the fourth row is the best choice and the only choice, because this choice is contained in all the other choices. If this choice fails, no other choice can success. Also, any other choice will assign one unnecessary input value which may cause more state requirements and generate longer test length. Without looking the whole table, the information column in the ON-set is not sufficient to avoid wrong choices. However, wrong choices can be avoided with the information column in the OFF-set. Since the output is value 1, all the positive numbers in the OFF-set are changed to be negative. Then, the third row will have value -1, which means input *f* cannot have value 0 otherwise conflicts will occur. Therefore, input *f* is assigned value 1, and hence the output value request is satisfied by the fourth row in the ON-set without any decision making.

Table VI.
The modified OFF-set of Figure 2 after a=1.

a	b	c	d	
0	1	x	x	&
1	0	x	x	1
0	x	1	x	&
x	1	0	x	2
x	x	0	1	2
x	x	1	0	2
				4

Table VII.
ON-set after c=1

a	b	c	d	e	f	
1	0	0	1	0	x	&
x	x	x	x	1	1	2
x	x	x	0	x	1	2
x	x	1	x	x	1	1
x	0	x	x	x	1	2
1	x	x	x	x	1	2
						5

Table VIII.
OFF-set after c=1

a	b	c	d	e	f	
x	x	x	x	1	0	2
x	x	x	0	x	0	2
x	x	1	x	x	0	1
x	1	x	x	x	0	2
0	x	x	x	x	0	2
0	1	0	1	0	x	&
						5

In summary, with the information column, justification and backward implication are done by looking up the information column to find a choice and the row choice chosen; while forward implication is done by looking up the column under the input with value assigned and the information column. In this manner, justification and implication for Boolean equations are done efficiently without redundant table look-up.

For multiple output functional blocks, we may find some outputs share same row choices. Test generation time and memory requirement can be reduced if we can combine their ON-sets and OFF-sets together. The combined ON-sets and OFF-sets will be similar to the personality matrices of PLAs.

III. EXPERIMENTAL RESULTS

We did an experiment on a highly sequential circuit which has 4 functional blocks in functional level and has 4 inputs, 6 outputs, and 110 gates and 14 flip-flops in gate level. Three kinds of circuit description were used as shown in Table IX . The experiment was done on CONVEX C-1. The unit of time is second. No fault simulator was used in this experiment. For every nonequivalent fault, HiSTG was tried to find a test sequence. The initial state was assumed to be unknown for every target fault. The efficiency is defined as the sum of detected faults and untestable faults divided by the total faults. Using multiple levels description, the total time is reduced about 50% compared with the time spend while using gate-level description. The functional block described in Tables VII and VIII is contained in this circuit. Using two tables to achieve better backward implication in HiSTG did reduce significantly the number of total vectors. Using functional-level description, HiSTG has the best performance, but only pin faults and the faults at delay elements of these four functional blocks are included in the fault list. It will be interesting to find the fault coverage of the vectors generated in each case. The built-in fault simulator in HiSTG is still under development. We would like to report the results as soon as possible.

Table IX.

description	multiple	functional	gate
cpu time	2362.98	329.14	3316.36
total faults	280	102	280
detected	223	92	221
untestable	31	7	29
dropped	26	3	30
efficiency	90.71%	97.06%	89.29%
vectors	5464	1166	9646

IV. CONCLUSIONS

The implementation of a hierarchical sequential test generator HiSTG was described. How to take advantages of Boolean equations to achieve high performance is also discussed. Due to the simplicity of the BACK algorithm and the SPLIT value model, the implementation of HiSTG is simple. The experimental results showed hierarchical test generation has three advantages: high performance, high fault coverage, and smaller test vectors.

REFERENCES

[1] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. Comput.*, Vol. C-16, pp. 567-579, Oct. 1967.

[2] G. R. Putzolu and J. P. Roth, "A Heuristic Algorithm for the Testing of Asynchronous Circuits," *IEEE Trans. Comput.*, Vol. C-20, pp. 639-647, June, 1971.

[3] P. Muth, "A Nine-valued Circuit Model for Test Generation," *IEEE Trans. Comput.*, Vol. C-25, pp. 630-636, June 1976.

[4] R. Marlett, "EBT, A Comprehensive Test Generation Technique for Highly Sequential Circuits," *15th Design Automation Conference,* Las Vegas, NV, pp. 250-256, June 1978.

[5] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. Comput.*, Vol. C-30, pp. 215-222, March 1981.

[6] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Comput.*, Vol. C-32, pp. 1137-1144, Dec. 1983.

[7] W. -T. Cheng, "The BACK Algorithm for Sequential Test Generation," to appear in *International Conference on Computer Design,* Rye Brook, New York, Oct. 1988.

[8] S. M. Thatte and J. A. Abraham, "Test Generation for Microprocessors," *IEEE Trans. Comput.,* Vol. C-29, pp. 429-441, June 1980.

[9] P. Bose and J. A. Abraham, "Test Generation for Programmable Logic Arrays," *19th Design Automation Conf.,* pp. 574-580, June 1982.

[10] M. A. Breuer and A. D. Friedman, "Functional Level Primitives in Test Generation," *IEEE Trans. Comput.,* Vol. C-29, pp. 223-234, Mar. 1980.

[11] Y. H. Levendel and P. R. Menon, "Test Generation Algorithms for Computer Hardware Description Languages," *IEEE Trans. Comput.,* Vol. C-31, pp. 577-588, July 1982.

[12] M. S. Abadir and H. K. Reghbati, "Functional Test Generation for Digital Circuits Described Using Binary Decision Diagrams," *IEEE Trans. on Computers,* Vol. C-35, pp. 375-379, April 1986.

[13] H. P. Chang, W. A. Rogers and J. A. Abraham, "Structured Functional Level Test Generation Using Binary Decision Diagrams," *International Test Conference,* pp. 97-104, 1986.

[14] F. Brglez, "A Hierarchical Testability Characterization: Formulation and Applications," *International Workshop on Logic Synthesis,* Research Triangle Park, North Carolina, May 12-15, 1987.

[15] S. J. Chandra and J. H. Patel, "A Hierarchical Approach to Test Vector Generation," *24th Design Automation Conference,* pp. 495-501, 1987.

[16] W. -T. Cheng, "SPLIT Circuit Model for Test Generation", *Proceedings of 25th Design Automation Conference,* pp. 96-101, June 1988.

[17] Private communications with Prof. Janak H. Patel, University of Illinois, Urbana-Champaign, Illinois, Feb. 1988.

ON THE DESIGN OF VLSI CMOS PLAs FOR BIST IMPLEMENTATION

R. DANDAPANI

Dept of ECE
University of Colorado
Colorado Springs, CO 80933

RAVI K. GULATI
DEEPAK K. GOEL
KENT R. CALLAHAN

Ford Microelectronics, Inc
10340 State Hwy 83
Colorado Springs, CO 80921

Abstract

Built-In Self Testing, Function-Independent Testable Programmable Logic Array (BIST FIT-PLA) designs are studied in this paper. We compare and contrast two recently proposed approaches. A dynamic CMOS implementation of PLAs is presented. We show that all single stuck-at, bridging, crosspoint and stuck-open faults, and almost all multiple stuck-at, bridging, and crosspoint faults are detected in our implementation of the two BIST methods. These methods are not directly applicable to folded PLAs. We present a method for designing BIST folded FIT-PLAs.

1. INTRODUCTION

Function-independent testable (FIT) Programmable Logic Array (PLAs) are of current importance as they are well suited for Built-In Self-Testing (BIST). The FIT-PLAs are designed by adding extra product and sum lines to a given PLA so as to generate odd parity at the outputs [1-8]. The BIST FIT-PLA of [6] was the culmination of design for testability efforts applicable to nMOS PLAs. The universal test set in [6] will detect all single and almost all multiple crosspoint, bridging and stuck-at faults in the two arrays of the FIT-PLA. But many faults in the extra circuit to control the application of the universal test set are not detected. The presence of some of these undetected faults could invalidate the universal test set. The FIT-PLA design of [7] follows the approach of [6] but provides tests for the added circuitry also.

The universal test set, T, of [6] is an ordered set of sequences. As a result of the ordering in T only one observation is sufficient on chip for a GO/NOGO indication. This method requires that the output signals be XORed to generate a parity signal. This causes a delay which could result in testing being done at a rate slower than system speed for PLAs with large number of outputs. On the other hand, there is no need to store and evaluate the intermediate outputs of the sequences. A design was proposed in [8] which uses the same test set T but avoids the need for XOR gates. This is achieved by requiring $2(n+1)$ observations, where n is the number of inputs to the PLA. As the number of observations is dependent on the number of inputs, it is not easy to evaluate the outputs on chip. The method of [8] is therefore suited for PLAs embedded on chips designed with scan path. However, only $2(n+1)$ scan-outs are used for the test set of size $2m(n+1)+1$, where m is the number of product terms. That is, hardware (internal to the chip) is needed to accumulate parity, and to control the times when scan out can occur. This is in contrast to normal scan designs where scan out occurs at constant time intervals. The advantage of the method is that testing may be performed at a rate closer to system speed. It was shown in [8] that T will detect CMOS specific stuck-open faults in their design. This is in addition to the detection of crosspoint, bridging and stuck-at faults.

In this paper we first analyze the methods of [6] and [8]. The two methods use identical test sets and differ only in the manner in which output parities are generated and the extra lines are personalized. It will also be shown that the method of [8] is incomplete as given in their paper and requires additional observations to satisfy the claims. We use a dynamic CMOS PLA implementation that has some differences with that of [8]. It will be shown that the universal test set will detect crosspoint, bridging, stuck-at and CMOS stuck-open faults for our implementation also. The methods of [6] and [8] are for non-folded PLAs only. We give a BIST design method for folded CMOS PLAs by extending the approach of [6] (and [8]).

The design procedures of [6] and [8] are compared in Section 2. A dynamic CMOS implementation and the justifications that the universal test set of [6] will detect the single stuck-open faults along with detecting all single and almost all multiple crosspoint, bridging and stuck-at faults are given in Section 3. In Section 4, we give the design of FIT-PLA with folding. Conclusions are given in Section 5.

2. PREVIOUS WORK

A PLA is given in Figure 1. This is the example used in [8]. It has 3 inputs, 2 outputs and 3 product terms. The FIT-PLA resulting from the method of [6] is given in Figure 2. Two shift registers are used: one for controlling the inputs (IS) and the other for controlling the product terms (PS). The IS register needs the following control inputs: (i) to set all stages to logic 1, (ii) to set IS1 to logic 1 and reset all other stages, (iii) to reset all stages, (iv) to reset IS1 and set all other stages to logic 1, and (v) to shift. The PS register needs the following control inputs: (i) to reset all stages, (ii) to set PS1 to logic 1 and reset all other stages, and (iii) to shift. These control signals, and C1, C2 are generated by a Timing Controller. There is a cumulative parity toggle flip flop which is initially set to logic 0 and updated by the outputs generated due to each test vector. There is an expected value toggle flip-flop which is initially set to logic 1 and toggled at specific times during self test. Once the controller is put into test mode (by setting Mode input), it determines the instances when the expected value flip flop should be toggled and compared to cumulative parity flip flop. The inputs (I) to the controller come from PS and IS registers. If there is a difference in the cumulative parity and the corresponding expected response values then a fault is present in the PLA and this information must be kept until the end of the test. The output latch performs this task. It is initially set to logic 0, and once set to logic 1, due to a test, cannot be changed until the end of self test.

The test set T obtained for this FIT-PLA is given in Table 1(a). It has a total of 2m(n+1)+1 (where n and m are the number of inputs and product terms, respectively) or 33 test inputs, partitioned into sequences S1, S2, S3, S4 and S5. It is suggested in [6] to use C1=1, C2=IS1=IS2=IS3=0 for S1, and tests #18 through #33 to come prior to tests #2 through #17. These are alternative ways of arranging the tests which do not affect the cumulative parity generated. We have chosen the new sequencing as it is easier to compare with the method of [8]. The outputs resulting from applying the tests to the FIT-PLA of Figure 2 are given in Table 1(b). The column labeled P shows the parity generated at the outputs by each test, and the column labeled C shows the cumulative parity. The places where C is shown to have a value in the table indicate the times when expected response flip flop is toggled and compared with the value of C.

The FIT-PLA resulting from the method of [8] is given in Figure 3. As in the design of [6], two shift registers (IS and PS) are needed. These registers are controlled by signals similar to those in [6]. Unlike [6], the control signal to set IS1 is accomplished by providing an extra product line which has devices at all bit lines and whose output is fed to IS1. In this manner the IS register acts as a Nonlinear Feedback Shift Register generating the test inputs as desired. The necessary control signals, and C1, C2 are generated by a Timing Controller, just as in [6]. At the outputs a shift register is present, each stage of which also acts as a parity generator for the corresponding output.

The test set used by this method is the same as for [6], and is given in Table 1(a). The outputs generated are given in Table 1(c). The outputs that carry parity information as per [8], are boxed in Table 1(c). Thus, after application of sequences S1 and S2 (tests #1 through #5), registers OS1 and OS2 will contain the parity on z1 and z2, respectively. If the FIT-PLA is fault free then each of these parity bits will be one. To ascertain this fact, the testing will have to be suspended and the contents of OS register will have to be scanned out. There are 8, 2(n+1) in general, such scan outs required for the FIT-PLA. It is of interest to note in Table 1 that the P value in [6] is identical to the z3 output in [8], and the outputs z1 and z2 differ in only two places where observations are made (tests # 5 and # 21 in Table 1). This happens due to the addition of a device at the intersection of the extra product line S4 and the sum line z1, to have odd parity on z1 output.

As a point of information it should be noted that Lemma 2 of [8] claims that sequence S2 or S4 will detect all single crosspoint faults in the OR array for their dynamic CMOS FIT-PLA implementation. By observing Table 1(c) it is seen that the output of S2 or S4 sequence is not observed at z3 (the extra output line). That is, crosspoint faults on this output line are not tested by S2 or S4. For this example single crosspoint faults on the z3 output line are detected by the sequences S3 and S5. This is not true in general. Consider the PLA of Figure 4(a). Applying the method of [8] to this PLA we get the FIT-PLA given in Figure 4(b). Sequences S3 and S5 for this FIT-PLA do not detect the missing crosspoint fault at the intersection of p5 with z3 or p6 with z3. Thus Lemma 2 of [8], which claims, "... any combination of missing and extra crosspoint faults in the OR array that affects the parity of any output line will be detected," is incorrect. The only way these crosspoint faults will be detected is by observing the output z3 due to sequences S2 or S4. As per the design, the output on z3 due to either S2 or S4 will have an even number of ones. The method of [8] does not consider this extra output line during testing with sequences S2 or S4 as indicated by the statement, "After applying the entire set of m+2 patterns, an odd number of ones appear on each output line". Thus, [8] must modify their testing scheme to observe z3 during testing with S2 or S4 and verify that the resulting parity is even. If the resulting parity is odd then there is a fault in the FIT-PLA. This requirement of even parity on z3 is the opposite of expecting odd parity on outputs z1 and z2. Note that product lines p5 and p6 are identical in Figure 4(b). Both are, however, required to satisfy the requirement of odd number of devices and non-devices on z1.

Since test set T is identical for the methods of [6] and [8], the hardware used to generate T can be the same. In both designs the effect of the NOR gates controlling the bit lines can be implemented by pulldown devices at odd (even) numbered bit lines with their gate input connected to C1 (C2). IS, PS and MUXes at the inputs can all be implemented by either the method of [6] or [8]. There is one piece of extra hardware that will be needed for [8]. As can be seen from Figure 3, there is a device at the intersection of output line z1 and the extra product line p4. Thus, the method of [8] changes the functionality of the original PLA. To offset the effect of the added product line(s) it is essential to discharge them during normal operation. For the FIT-PLA of Figure 3, this can be accomplished by letting PS4 have the value of logic 0 when C1=C2=0 (the normal operating mode).

The two methods differ in the hardware required at the outputs. The relatively area intensive design of [6] is essential if the FIT-PLA is to be a BIST design in IC chips with no scan path. It gives a GO/NOGO output at the conclusion of the test. On the other hand the design of [8] saves area in the FIT-PLA at the cost of area required to implement scan path. There is a further complication in implementing this scheme in an existing scan based IC chip. Normal scan is implemented by requiring outputs of test results to be scanned out at constant time intervals. In [8] the results are scanned out only after the first m+1 patterns and after every m patterns thereafter. It is interesting to note that the method of [6] provides a better design for use in VLSI circuits with scan path. The outputs from Table 1(b) need not be computed on chip. The output of each test vector can be scanned out and evaluated outside the chip. This alternative is desirable to compressing the outputs on chip as suggested by [8].

A final note of comparison is the number of extra devices needed to implement the two methods. The difference in the numbers is obtained by considering the OR array only. The AND array is identical in both designs. Only one device was added in the OR array of Figure 2, whereas five devices were added in Figure 3. This disparity in the number of devices between the two methods is true in general. This point is mentioned here for completeness and will probably not affect the area overhead.

3. DYNAMIC CMOS PLA IMPLEMENATION

The PLA of Figure 1 implemented by us is given in Figure 5. It is a dynamic CMOS realization using two non-overlapping clocks Ø1 and Ø2. The AND and OR arrays are implemented using n-channel crosspoint devices. There is a separate pulldown n-channel device for each product and sum line. Also, each product and sum line is precharged by a p-channel (pullup) device. During Ø2, the product lines in the AND array are precharged by pullup devices PA and cannot be pulled down because the pulldown device for each product line is disabled. During Ø1, the evaluation phase, each product line is discharged only if one of the crosspoint devices on the product line is turned on. The outputs of the AND array are latched to permit the clocking scheme. The OR plane is precharged during Ø1, followed by the evaluation and latching of the outputs during Ø2.

It was shown in [6] that crosspoint, bridging and stuck-at faults in the AND and OR arrays of the nMOS FIT-PLA of [6] are detected by the universal test set. The proofs are based on the structure of the FIT-PLA and not on the technology used to implement it. In CMOS technology, it is possible to have bridging faults between (i) two precharge lines, (ii) precharge and pulldown lines, and (iii) two pulldown lines. The bridging fault between two precharge lines is analogous to the bridging fault considered in [6] for nMOS implementation. The bridging fault between precharge and pulldown lines is equivalent to a stuck-at-0 fault on the corresponding product or sum line, and is detected by the universal test set. A bridging fault between two pulldown lines is undetectable, but does not affect testing or operation of the PLA. Our FIT-PLA scheme is same as [6]. Thus the universal test set is directly applicable to our FIT-PLA.

Theorem 1: All single and almost all multiple crosspoint, stuck-at and bridging faults in our FIT-PLA are detected by the universal test set of [6].

CMOS stuck-open faults were not considered in [6] as the design was for nMOS technology. A stuck-open fault causes a transistor to remain non-conducting irrespective of the voltage at its gate terminal. For the FIT-PLA implementation of [8], correspondence was established between stuck-open faults and the conventional crosspoint or stuck-at faults, for devices in the AND and OR planes.

Our PLA implementation is similar to that of [8]. There is, however, one architectural difference. The scheme of [8] uses only one pulldown device for all the product lines and only one pulldown device for all the sum lines, whereas we use separate pulldown devices for each product (sum) line. It was found that using only one pulldown device for all the product (sum) lines requires an inordinately large device. Besides requiring more area, this gives rise to adverse charge injection problems, needing input drivers to do more work. Similar to [8], we establish the correspondence between stuck-open faults and crosspoint or stuck-at faults for our FIT-PLA implementation.

Regarding stuck-open faults, there are three classes of devices to consider. (1) Stuck-open fault in a crosspoint device is equivalent to a missing crosspoint fault in that device. (2) Stuck-open fault in a pulldown device NA (NO) is equivalent to a single product (sum) line stuck-at-1. To detect this stuck-open fault, the product (sum) line must be initialized to logic 1 prior to applying the test from sequences S3 and S5 (in the universal test set) used to detect the stuck-at-1 fault on it. The precharge phase satisfies the assertion to logic 1 and the test applied discharges the product (sum) line through the faulty device. As a point of information, for the implementation of [8] there is only one pulldown device for all product (sum) lines. Hence a stuck-open fault in the pulldown device for product (sum) lines is equivalent to a multiple stuck-at-1 fault on all product (sum) lines, whereas for our implementation, a stuck-open fault is equivalent to a single stuck-at-1 fault. (3) A stuck-open fault in the pullup device PA (PO) requires that the product (sum) line be initialized to logic 0, then charged to logic 1 to provoke the fault. If the stuck-open fault exists, the faulty product (sum) line will remain at logic 0. The first test in the universal test set discharges all the product lines. The next set of tests (sequence S2) place a logic 1 on the product lines, one at a time. Thus, a PA stuck-open fault will be detected. Note that the same series of tests also ensures that single PO stuck-open faults will be detected. We can now state the following result as a corollary to Theorem 1.

Corollary 1: Any combination of stuck-open faults that do not affect the parity of bit or sum lines will be detected by the universal test of [6].

In CMOS technology it is also possible to have stuck-on faults. Stuck-on fault detection require current measurements. We did not consider stuck-on faults in this study.

4. FOLDED FIT-PLA

In this section we consider the design of BIST folded PLAs. We allow bit and sum lines folding in the AND and OR arrays. The approach given here can be extended to product lines folding also, but is not given here.

Our approach to the design of BIST folded PLAs is as follows: Consider the non-folded version and apply the method of [6] or [8]. Up to two product lines will be added in this design. Add the same number of product lines to both sides of the AND array in the folded PLA. The added product lines in the folded PLA should contain devices at the intersection with bit lines as indicated in the non-folded design. The total number of product lines will depend on the number of product lines that are in the non-folded PLA. If the total number is odd then an additional product line, with no devices, is needed on one side of the AND array. All the added product lines are extended to the OR plane and personalized as in [6] or [8] ignoring the cut on sum lines, if any. The BIST folded PLA will have the following features:

(i) IS shift register will have the same number of stages as in the non-folded design, but will be distributed on both sides of the AND array.

(ii) PS register will have up to two extra stages compared to the non-folded design.

(iii) The parity will be computed by XORing the outputs from both sides of the OR array, if the design approach of [6] is desired. The number of stages in the OS register will be the same as in non-folded design, if the approach of [8] is desired. But the register will be distributed on both sides of the OR array.

The construction of BIST folded PLAs is demonstrated with the aid of the examples given in Figures 6 and 7. We give the modifications to the OR plane with respect to the design approach of [6] only. Similar modifications to the OR plane can be made for the approach of [8]. Figure 6 (Figure 7) is an example where the given unfolded PLA has an even (odd) number of product terms.

The methods of [6] and [8] for non-folded PLAs result in a BIST PLA which has an even number of product lines. Thus the product line shift register has an even number of stages. Also, the resultant number of non-devices on each bit line is odd. Due to these reasons, the parity generated will be odd for each cycle of the product line shift register while testing with the sequences S3 and S5. With the design approach given above for folded PLAs, we have retained the above testing approach of non-folded PLAs. That is, while a bit line is at logic 1, only an odd number of product lines will be pulled low by circulating a logic 1 in the PS register. Note that, while testing with S2 or S4, the number of devices and non-devices in the AND array do not affect the parity, as the bit lines are all at logic 0.

An universal test set, Tf, similar to T for non-folded PLAs, can be derived. Due to the increase in the size of PS shift register, Tf will have up to 4(n+1) extra tests compared to T. As in [6] or [8], it can be shown that all single stuck-at, crosspoint, bridging and stuck-open faults, and almost all multiple stuck-at, crosspoint and bridging faults will be detected by the test sequence Tf.

The product line added to make the total lines even (p6 in Figure 7) will have only one device, at the intersection with the extra sum line. It is possible to eliminate this product line and XOR the corresponding stage of PS shift register with the parity generated by the output lines in the approach of [6].

5. CONCLUSIONS

In this paper we have studied the problem of designing BIST FIT-PLAs. Two approaches, due to [6] and [8], were considered in detail. Both the methods are structural, function independent testable and have analytically proved fault coverage.

The method of [6] requires only one observation at the end of self test. This can be generated on chip. The method of [8] requires observations depending upon the number of inputs. Hence this method is suitable for VLSI circuits with scan path. It was shown that the method of [8] is incomplete as given in the paper and requires some adjustments in the output observations. It was also established that the method of [8] is not much different from the method of [6].

We presented an implementation of dynamic CMOS PLAs. This is similar to the implementation of [8] with slight differences. It was shown that the universal test set of [6] (hence [8]) is directly applicable to our implementation also.

The folded FIT-PLA was studied and a method to design BIST folded PLAs was proposed. The design approach results in the testing time being slightly more than the testing time for the non-folded version. The fault coverage is all single and almost all multiple stuck-at, crosspoint, bridging faults and all single stuck-open faults.

The input buffers, AND plane output latches, and OR plane output latches have been assumed fault free. Also, Theorem 1 and Corollary 1 do not address the testability of the test circuitry. These issues are under investigation.

REFERENCES

[1] S.J. Hong and D.L. Ostapko, "FIT-PLA: A programmable Logic Array for Functional Independent Testing," *Dig. 10th Int. Symp. Fault-Tolerant Comput.*, pp. 131-136, 1980.

[2] S. Yajima and T. Aramaki, "Autonomously Testable Programmable Logic Arrays," *Dig. 11th Int. Symp. Fault-Tolerant Comput.*, pp. 41-43, 1981.

[3] H. Fujiwara and K. Kinoshita, "A Design of Programmable Logic Arrays with Universal Tests," *IEEE Trans. Comput.*, Vol. C-30, pp. 823-828, Nov. 1981.

[4] H. Fujiwara, "A New PLA Design for Universal Testability," *IEEE Trans. Comput.*, Vol. C-33, No. 8, pp. 745-750, Aug. 1984.

[5] K. A. Hua, J. -Y. Jou, and J. A. Abraham, "Built-In Tests for VLSI Finite State Machines," *Dig. 14th Int. Symp. Fault-Tolerant Comput.*, pp. 292-297, 1984.

[6] R. Treuer, H. Fujiwara, and V.K. Agrawal, "Implementing a Built-In Self-Test PLA Design," *IEEE D & T*, pp. 37-48, April 1985.

[7] D. S. Ha and S. M. Reddy, "On BIST PLAs," *Proc. Int. Test Conf.*, pp. 342-351, 1987.

[8] D. L. Liu and E. J. McCluskey, "Design of Large Embedded CMOS PLA's for Built-In Self-Test," *IEEE Trans. CAD,* Vol 7, No. 1, pp. 50-59, Jan. 1988.

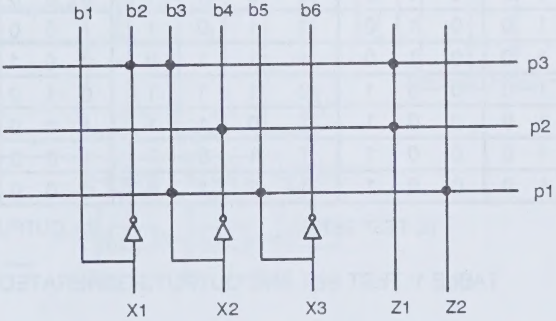


Figure 1 : A PLA

SEQ	#	C1	C2	IS1	IS2	IS3	PS1	PS2	PS3	PS4	z1	z2	z3	P	C	z1	z2	z3
S1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
S2	2	0	1	1	1	1	0	1	1	1	0	1	0	1	1	0	1	1
	3	0	1	1	1	1	1	0	1	1	1	0	0	1	0	1	0	1
	4	0	1	1	1	1	1	1	0	1	1	0	0	1	1	1	0	1
	5	0	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	1
S3	6	0	1	0	1	1	0	1	1	1	0	1	0	1		0	1	1
	7	0	1	0	1	1	1	0	1	1	1	0	0	1		1	0	1
	8	0	1	0	1	1	1	1	0	1	1	0	0	1		1	0	1
	9	0	1	0	1	1	1	1	1	0	0	0	0	0	1	0	0	0
	10	0	1	1	0	1	0	1	1	1	0	0	0	0		0	0	0
	11	0	1	1	0	1	1	0	1	1	1	0	0	1		1	0	1
	12	0	1	1	0	1	1	1	0	1	0	0	0	0		0	0	0
	13	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	14	0	1	1	1	0	0	1	1	1	0	0	0	0		0	0	0
	15	0	1	1	1	0	1	0	1	1	1	0	0	1		1	0	1
	16	0	1	1	1	0	1	1	0	1	1	0	0	1		1	0	1
	17	0	1	1	1	0	1	1	1	0	0	0	1	1	1	1	0	1
S4	18	1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	1	1
	19	1	0	0	0	0	1	0	1	1	1	0	0	1	1	1	0	1
	20	1	0	0	0	0	1	1	0	1	1	0	0	1	0	1	0	1
	21	1	0	0	0	0	1	1	1	0	0	0	1	1	1	1	0	1
S5	22	1	0	1	0	0	0	1	1	1	0	1	0	1		0	1	1
	23	1	0	1	0	0	1	0	1	1	1	0	0	1		1	0	1
	24	1	0	1	0	0	1	1	0	1	0	0	0	0		0	0	0
	25	1	0	1	0	0	1	1	1	0	0	0	1	1	0	1	0	1
	26	1	0	0	1	0	0	1	1	1	0	1	0	1		0	1	1
	27	1	0	0	1	0	1	0	1	1	0	0	0	0		0	0	0
	28	1	0	0	1	0	1	1	0	1	1	0	0	1		1	0	1
	29	1	0	0	1	0	1	1	1	0	0	0	1	1	1	1	0	1
	30	1	0	0	0	1	0	1	1	1	0	1	0	1		0	1	1
	31	1	0	0	0	1	1	0	1	1	1	0	0	1		1	0	1
	32	1	0	0	0	1	1	1	0	1	1	0	0	1		1	0	1
	33	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0

(a) TEST SET T (b) OUTPUT [6] (c) OUTPUT [8]

TABLE 1: TEST SET AND OUTPUTS GENERATED

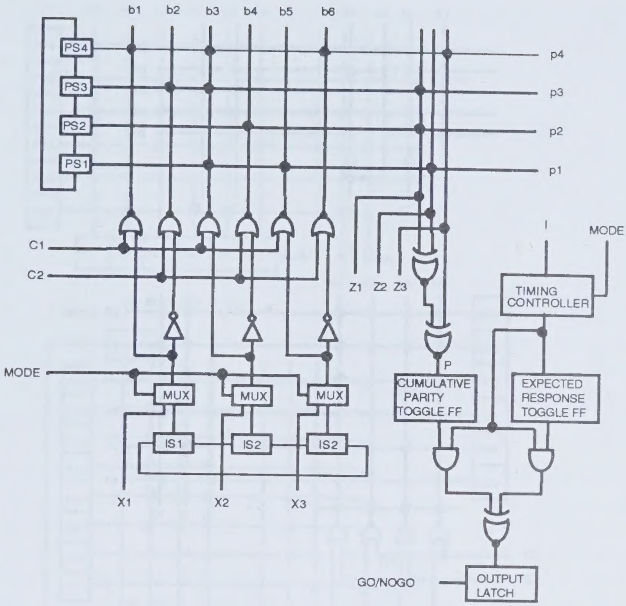

b1
b2
b3
b4
b5
b6
PS4
PS3
PS2
PS1
p4
p3
p2
p1
C1
C2
MODE
MUX
MUX
MUX
IS1
IS2
IS2
X1
X2
X3
Z1
Z2
Z3
P
I
MODE
TIMING CONTROLLER
CUMULATIVE PARITY TOGGLE FF
EXPECTED RESPONSE TOGGLE FF
GO/NOGO
OUTPUT LATCH

Figure 2 : BIST FIT PLA of Figure 1 due to [6]

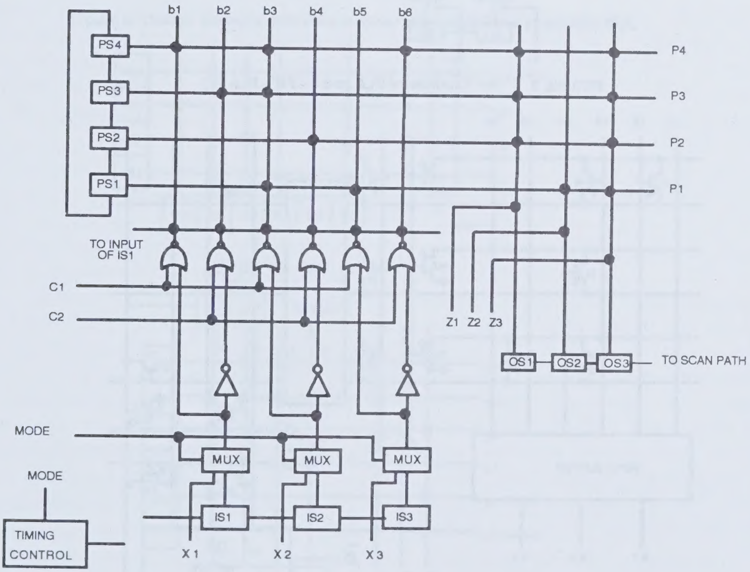
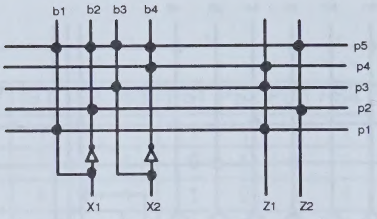
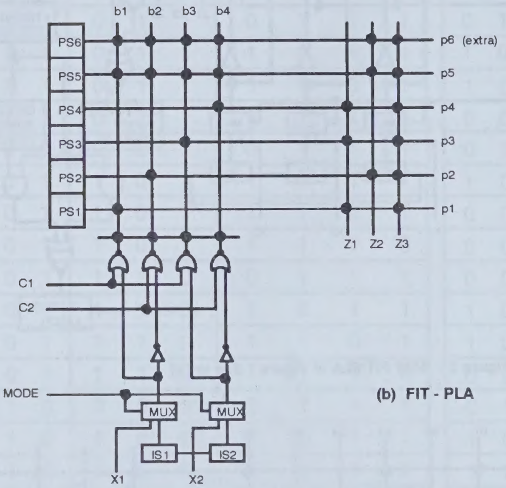

b1
b2
b3
b4
b5
b6
PS4
PS3
PS2
PS1
P4
P3
P2
P1
TO INPUT OF IS1
C1
C2
Z1
Z2
Z3
OS1
OS2
OS3
TO SCAN PATH
MODE
MUX
MUX
MUX
MODE
TIMING CONTROL
IS1
IS2
IS3
X 1
X 2
X 3

FIGURE 3 FIT - PLA of Figure 1 due to [8]


b1
b2
b3
b4
p5
p4
p3
p2
p1
X1
X2
Z1
Z2

(a) A PLA


b1
b2
b3
b4
PS6
PS5
PS4
PS3
PS2
PS1
p6 (extra)
p5
p4
p3
p2
p1
Z1
Z2
Z3
C1
C2
MODE
MUX
MUX
IS1
IS2
X1
X2

(b) FIT - PLA

FIGURE 4 : An Example of PLA and its FIT - PLA [8]

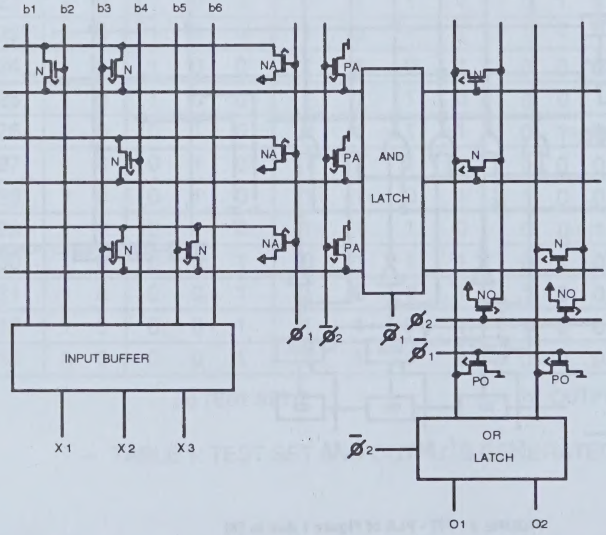
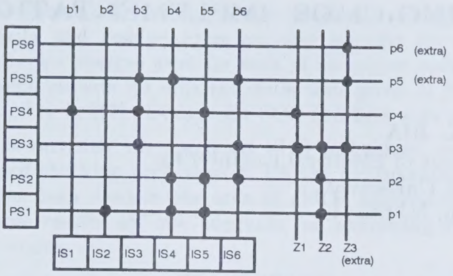
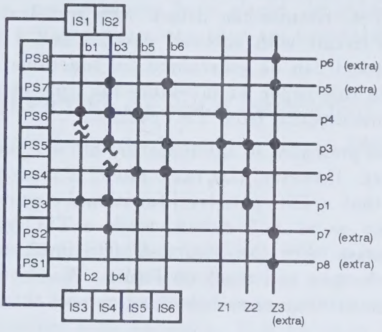

b1
b2
b3
b4
b5
b6
N
N
N
N
N
NA
NA
NA
PA
PA
PA
AND
LATCH
N
N
N
NO
NO
PO
PO
INPUT BUFFER
X 1
X 2
X 3
ϕ_1
$\bar{\phi}_2$
$\bar{\phi}_1$
ϕ_2
$\bar{\phi}_1$
$\bar{\phi}_2$
OR
LATCH
O1
O2

FIGURE 5 : DYNAMIC CMOS IMPLEMENTATION OF PLA OF FIGURE 1

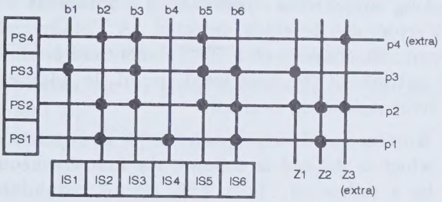

b1 b2 b3 b4 b5 b6
PS6
PS5
PS4
PS3
PS2
PS1
p6 (extra)
p5 (extra)
p4
p3
p2
p1
IS1 IS2 IS3 IS4 IS5 IS6
Z1 Z2 Z3
(extra)

(a) BIST PLA of [6]

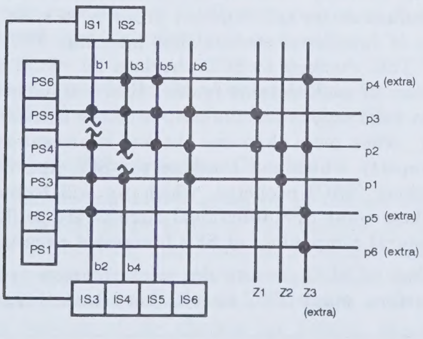

IS1 IS2
b1 b3 b5 b6
PS8
PS7
PS6
PS5
PS4
PS3
PS2
PS1
p6 (extra)
p5 (extra)
p4
p3
p2
p1
p7 (extra)
p8 (extra)
b2 b4
IS3 IS4 IS5 IS6
Z1 Z2 Z3
(extra)

(b) BIST folded PLA

Figure 6: Design example with even number of product lines in original PLA.


b1 b2 b3 b4 b5 b6
PS4
PS3
PS2
PS1
p4 (extra)
p3
p2
p1
IS1 IS2 IS3 IS4 IS5 IS6
Z1 Z2 Z3
(extra)

(a) BIST PLA of [6]


IS1 IS2
b1 b3 b5 b6
PS6
PS5
PS4
PS3
PS2
PS1
p4 (extra)
p3
p2
p1
p5 (extra)
p6 (extra)
b2 b4
IS3 IS4 IS5 IS6
Z1 Z2 Z3
(extra)

(b) BIST folded PLA

Figure 7 : Design example with odd number of product lines in original PLA.

SFS/SSC DOMINO-CMOS IMPLEMENTATIONS OF TSC CIRCUITS

NIRAJ K. JHA
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

ABSTRACT

The totally self-checking (TSC) concept is well-established for applications in the area of on-line error-indication. TSC circuits can detect both transient and permanent faults. They consist of a functional circuit with encoded outputs and a checker which monitors these outputs. The TSC concept can be generalized for functional circuits to the strongly fault secure (SFS) concept. In this paper we introduce the concept of strongly self-checking (SSC) checkers, which is a generalization from TSC checkers.

Most of the TSC circuits presented in the literature are designed at the logic gate level using the stuck-at fault model. However, this fault model is inadequate for MOS technologies. In this paper we show that a TSC gate-level functional circuit can be implemented in the domino-CMOS technology as an SFS circuit, while a TSC gate-level checker can be implemented as an SSC checker. For the domino-CMOS implementation we enlarge the fault model to stuck-at, stuck-open and stuck-on faults. We show that domino-CMOS is much more suitable for implementation of self-checking circuits than static CMOS.

1. INTRODUCTION

In VLSI circuits most of the errors are transient in nature [1]. Off-line testing is obviously inadequate for detecting such errors. However, if Concurrent Error Detection (CED) circuits are used, transient errors can be easily detected. A Totally Self-Checking (TSC) circuit is one such CED circuit. It consists of a TSC functional circuit and a TSC checker. The checker monitors the outputs of the functional circuit. In addition, even if the checker itself develops a fault, an error indication is given.

The concept of TSC circuits was first proposed in [2] and generalized in [3]. These circuits meet the TSC goal, which is defined as follows: the first erroneous output of the functional circuit should not be a codeword. In a TSC circuit redundancies are not allowed. The design is such that every modeled fault is detected by an input vector. Smith generalized the TSC concept to the Strongly Fault Secure (SFS) concept for functional circuits [4]. SFS circuits can tolerate redundancies and function properly in spite of undetectable faults. They are the largest class of functional circuits that meet the TSC goal. Nicolaidis et al generalized the concept of TSC checkers to SCD checkers [5]. SCD checkers can also function properly in the presence of undetectable faults. However, as will be shown later, the concept of SCD checkers is valid only when the output of the checker is directly observable. When this is not true (e.g., when more than one checker is present in the self-checking system), we need another property which can combine the SFS and SCD concepts. We call this the strongly self-checking (SSC) property, which we will formally define later. SSC checkers are to TSC checkers what SFS functional circuits are to TSC functional circuits. Thus, SSC checkers are natural companions of SFS functional circuits.

It has been found that in VLSI circuits the most common type of error is unidirectional in nature [6]. Therefore, many TSC circuits have been designed to protect against

---

Acknowledgments: This work was supported in part by Siemens Corporation under Grant 170-4051 and in part by National Science Foundation under Grant MIP-8708728.

such errors. Some of the commonly used codes are the m-out-of-n code, the Berger code and the two-rail code. Parity and residue codes are also popular for detecting single errors. Many TSC gate-level checker designs exist for each of the above codes [7]-[15]. Methods for obtaining TSC functional circuits for different codes were given in [16]. A method for realizing an SFS PLA was given in [17]. Designs for SCD checkers were given in [5].

Although CMOS is currently the dominant VLSI technology, not much work has been done in verifying the self-checking properties of CMOS implementations of TSC gate-level circuits. Some work has been done in the area of nMOS implementations of TSC circuits [18]-[20]. However, these results are not adequate for addressing the problems that arise with CMOS implementations.

In this paper we will concentrate on domino-CMOS implementations of TSC gate-level functional circuits and checkers. We will establish the self-checking properties of these implementations under a comprehensive fault model, which includes stuck-at, stuck-open and stuck-on faults. We will also show why they are preferable to static CMOS implementations.

2. PROPERTIES OF SELF-CHECKING CIRCUITS

In this section we will discuss the properties that a TSC, SFS, SCD or SSC circuit must have.

A. Definitions

The following definitions can be used to define a TSC system [2], [3].

Definition 1: A circuit G is *self-testing* with respect to a set of faults F if for every fault in F the circuit produces a non-code output for at least one code input.

Definition 2: G is *fault-secure* with respect to F if for every fault in F the circuit never produces an incorrect code output for any code input.

Definition 3: G is *totally self-checking (TSC)* if it is both self-testing and fault-secure.

Definition 4: G is *code-disjoint* if it always maps code inputs to code outputs and non-code inputs to non-code outputs.

Definition 5: G is a *TSC checker* if it is TSC and code-disjoint.

The following definitions can be given for SFS and SCD circuits. They are equivalent to the ones given in [4] and [5].

Definition 6: G is *strongly fault secure (SFS)* with respect to F if for every fault in F, either

(a) the circuit is TSC, or

(b) the circuit is fault-secure, and if another fault from F occurs in the circuit then either property (a) or (b) is true for the multiple fault.

Definition 7: G is *strongly code disjoint (SCD)* with respect to F if before the occurrence of any fault, G is code-disjoint, and for every fault in F, either

(a) the circuit is self-testing, or

(b) the circuit always maps noncode inputs to noncode outputs, and if another fault from F occurs then either property (a) or (b) is true for the multiple fault.

B. Role of the Fault-Secure Property in a Checker

There has been an interesting debate going on in the self-checking community about the usefulness of the fault-secure property for a TSC checker. It was mentioned in [5] that the fault-secure property is not really important in a checker. The reason given was that since a code or noncode output from a checker respectively indicate correct or incorrect operation of the system, it does not matter which particular code output is produced. This reasoning is valid if the outputs of the checker are directly observable. However, as has been pointed out in [21], a system usually consists of many checkers whose outputs are checked by a final two-rail checker. For the final checker the fault-secure property is not needed. However, for the other checkers the fault-secure property is still required, otherwise the final checker may not get its test set and, hence, may not be self-testing.

We will now extend the above arguments to the SCD checker. From Definition 7 it is clear that the SCD property is defined for a checker whose outputs are observable, in other words, for the final checker. However, for the other checkers in the self-checking system we need the SFS property as well. The SFS and SCD properties can be combined in the following definition.

Definition 8: G is *strongly self-checking (SSC)* with respect to F if before the occurrence of any fault, G is code-disjoint, and for every fault in F, either

(a) the circuit is self-testing and fault-secure, or

(b) the circuit is fault-secure and always maps noncode inputs to noncode outputs, and if another fault from F occurs then either property (a) or (b) is true for the multiple fault.

Some assumptions are usually made for self-checking circuits. Faults are assumed to occur one at a time. After the occurrence of a fault in the circuit, it is assumed that sufficient time elapses such that all its required tests (codewords) are applied before a new fault can occur. It can be easily shown that a self-checking system consisting of a TSC or SFS functional circuit and a TSC or SSC checker achieves the TSC goal under the above assumptions.

For the gate-level TSC circuits the stuck-at fault model is usually assumed. However, this fault model has been shown to be inadequate for MOS technologies [22], [23]. For our domino-CMOS implementations we will assume the following fault models: stuck-at, stuck-open and stuck-on. These fault models cover most of the realistic physical failures that occur in a MOS circuit.

3. WHY USE DOMINO-CMOS, NOT STATIC CMOS ?

In this section we will establish the superiority of domino-CMOS over static CMOS for implementing self-checking circuits. This is best illustrated through an example.

In the following sub-sections we will discuss the gate-level, domino-CMOS and static CMOS implementations respectively of a two-rail checker.

A. Gate-Level Two-Rail Checker

Consider the two-variable two-rail checker, denoted TRC_2, in Fig. 1. In this checker (x_1x_2) forms one rail of the code and (y_1y_2) the second rail, which carries logic values complement to those on the first rail. The codeword set C on lines x_1,y_1,x_2,y_2 is {0101, 0110, 1001,1010}. It can be easily shown that this set detects all single stuck-at faults in TRC_2.

B. Domino-CMOS Two-Rail Checker

We now consider the domino-CMOS implementation in Fig. 2 of the sub-circuit of TRC_2 with output z_1. The implementation consists of an nMOS network consisting of transistors T_1-T_4 which implements the logic, a clocked pMOS transistor P1 used for precharging node $\overline{z}_1$, a clocked nMOS transistor N1 used in the evaluation phase, and an inverter to realize the function z_1. For simplicity we refer to a node by the same name as that of the function realized at that node. When clock=0, $\overline{z}_1$ is precharged to logic 1. When clock=1, $\overline{z}_1$ remains at logic 1 or discharges to logic 0 depending on whether there is a conduction path through the nMOS network. The new inputs arrive at the circuit during the precharge phase. The evaluation phase starts only after the inputs have settled down. Therefore, timing skews in the variable changes can not cause a glitch at $\overline{z}_1$ or z_1.

We can see that all the single stuck-at, stuck-open and stuck-on faults in the nMOS network can be detected by the codeword set C. In fact a stuck-at 1 (stuck-at 0) fault at the input of transistor T_i, $1 \le i \le 4$, is equivalent to a stuck-on (stuck-open) fault in T_i. C also detects a stuck-open fault in N1 or P2. However, to detect a stuck-open fault in N2 or P1 a two-pattern test is required. This consists of an initialization input followed by a test input. The initialization input can be any codeword which makes $z_1 = 1$ (i.e., 0110 or 1001). The test input can be any codeword which makes $z_1 = 0$ (i.e., 0101,1010). Hence, there are four different ways in which these stuck-open faults can be detected.

Before we consider the stuck-on faults in transistors N1, P1, N2 and P2, the following definition is needed.

Definition 9: A CMOS gate is n-dominant (p-dominant) if, when both its nMOS and pMOS networks conduct, the nMOS (pMOS) network dominates and the output of the gate is logic 0 (1).

Assume that both the complex gate and the inverter in Fig. 2 are p-dominant. Therefore, a stuck-on fault in P1 (P2) would become equivalent to a stuck-at 1 fault at $\overline{z}_1$ (z_1), and are hence detectable. The stuck-on faults in N1 or N2 are not detectable using logic monitoring. However, presence of these faults in the circuit does not affect the self-checking property of the circuit, as will be shown in Section 4. Note that usually current monitoring is done in CMOS circuits to detect stuck-on faults. The reasoning behind this method is that when a test is applied for a stuck-on fault, both the networks conduct and a much larger than normal current is drawn. Thus, N1 or N2 stuck-on fault would be detectable using current monitoring. However, although current monitoring is a viable method for flushing out stuck-on faults at the time of testing newly manufactured chips, it is not practical to incorporate it in the self-checking environment. Current monitoring is much slower than logic monitoring and, hence, is not very useful in an assembled self-checking system. Therefore, it is reasonable to assume that only logic monitoring is possible in on-line error detection circuits.

Finally, a stuck-at fault on the remaining lines (i.e., the clock line, $\overline{z}_1$,z_1, etc) can be shown to be easily detectable. Similarly, all the above arguments can also be applied to the domino-CMOS implementation of the subcircuit realizing z_2 in Fig. 1.

C. Static CMOS Two-Rail Checker

Let us now look at the static CMOS implementation in Fig. 3 of the subcircuit of TRC$_2$ realizing z_1. In this circuit every stuck-open in T_1–T_8 and the two transistors in the inverter requires a two-pattern test. During normal operation even if all the codewords from C are applied to this circuit many times, there is no guarantee that all the required two-pattern tests will be applied. Therefore, the assumptions required for TSC circuits, which are given in Section 2.B, may be violated, because it assumes that every detectable fault is detected before the next fault occurs. The above problem becomes more severe as the size of the circuit increases.

Another problem with static CMOS circuits is test set invalidation due to circuit delays and timing skews in input changes [24]. Test sets which cannot be invalidated are called robust test sets. For some static CMOS circuits robust test sets do not exist. These circuits have to be redesigned for testability [24]-[28].

Consider the circuit in Fig. 3. again. The two-pattern test <0110, 0101> should detect the stuck-open faults in transistors T_5 and T_8. However, in the presence of delays, x_2 may change from 1 to 0 before y_2 changes from 0 to 1. Therefore, this two-pattern test cannot robustly detect the stuck-open fault in T_5. This fact illustrates why it is even more difficult, if at all possible, to satisfy the assumptions required for TSC circuits for their CMOS implementations in general. Since only codewords are available for detecting faults in the self-checking circuits, robust stuck-open fault detection is frequently not possible. Therefore, redesigning these circuits for testability becomes a necessity.

Let us now consider a stuck-on fault in transistor T_7 in Fig. 3. Its only possible codeword test is 0110. However, since both networks will conduct when this test is applied the logic value at $\overline{z}_1$, will be 1 or 0 or in between, depending on whether the pMOS or the nMOS network or neither dominates. If the nMOS network dominates then T_7 stuck-on fault is undetectable. The presence of an undetectable fault may prevent other faults from being detected. For example, the two-pattern test <0110, 0101> can no longer detect T_8 stuck-open when T_7 stuck-on is already present. Furthermore, a non-codeword at the checker inputs may map to a codeword at its outputs in the presence of the undetected faults, thus destroying its code-disjoint property. Hence, it is clear that making the implementation n-dominant or p-dominant does not solve the problem. If the circuit is made n-dominant (p-dominant) the stuck-on faults in the pMOS (nMOS) network can no longer be detected. These undetected faults may destroy the code-disjoint property.

The advantages of the domino-CMOS implementation over the static CMOS implementation of a self-checking circuit can be summarized as follows:

(1) A more comprehensive fault model, which includes stuck-at, stuck-open and stuck-on faults, can be assumed for a domino-CMOS implementation, whereas for a static CMOS implementation stuck-on faults cannot be included.

(2) Every stuck-open fault in a static CMOS implementation requires a two-pattern test. Furthermore, to ensure robust detection of all stuck-open faults the circuit may have to be redesigned for testability. However, there are very few stuck-open faults in a domino-CMOS implementation that require a two-pattern test. All these faults can be robustly detected and no redesign is needed.

(3) For large static CMOS implementations, the assumptions required for TSC circuits are not easy to satisfy. If these basic assumptions are not met then the implementation cannot be guaranteed to be self-checking even if only stuck-at and stuck-open faults are assumed to occur. For the domino-CMOS implementations these assumptions are realistic even though the fault model also includes stuck-on faults.

The disadvantage of a domino-CMOS implementation is that only non-inverting functions can be implemented with it. For many self-checking circuits this does not create any problems as they do not contain any inverters. However, for some self-checking circuits this may be a problem. Such circuits can be implemented with another dynamic CMOS circuit implementation - cascode voltage switch logic (CVSL) [29]. These issues are discussed in more detail in the next section.

4. SELF-CHECKING DOMINO-CMOS IMPLEMENTATIONS

A self-checking system consists of a functional circuit and a checker. In this section we will start with the gate-level implementations of the functional circuit and the checker, which are TSC with respect to stuck-at faults. We will show that the domino-CMOS implementation of the functional circuit (checker) is Strongly Fault Secure (Strongly Self Checking) with respect to stuck-at, stuck-open and stuck-on faults.

The pioneering work in the area of TSC gate-level functional circuit design was done by Smith [16]. He showed that inverter-free TSC gate-level realizations are possible to obtain for functional circuits whose outputs are encoded with an all-unidirectional error-detecting code, such as the m-out-of-n code or the Berger code. An inverter-free gate-level realization can be easily implemented using the domino-CMOS technique. Similarly the TSC gate-level realizations of many checkers are also inverter-free [7]-[12], [15]. They are also amenable to domino-CMOS implementation. However, some gate-level TSC circuits may not be inverter-free. For such circuits we can use Cascode Voltage Switch Logic (CVSL) implementations. These implementations are considered in another paper [30]. In this paper we assume that the given TSC gate-level realization is inverter-free, so that a domino-CMOS implementation is possible.

In a domino-CMOS circuit every complex gate is followed by an inverter. We will refer to the two gates as a gate-pair. For example, the circuit in Fig. 2 is a gate-pair. We will assume that every complex gate in the domino-CMOS implementation is made p-dominant, and the inverter is either n-dominant or p-dominant. It is quite easy to meet this assumption in practice.

A. Illustration of Technique Through an Example

Our proof is best illustrated by an example first. We will formalize it later. Consider the circuit in Fig. 2 again. Assume first that both the complex gate and inverter are p-dominant, as before. Therefore a stuck-on fault in transistor N1 or N2 would be undetectable. Our aim will be to show that these undetectable faults do not destroy the self-checking property. The following cases arise.

Case 1: fault in the nMOS network of the complex gate.

A stuck-open (stuck-on) fault in a transistor in the nMOS network (i.e., in $T_1 - T_4$) is equivalent to a stuck-at 0 (stuck-at 1) fault at the input of the transistor. Since every single stuck-at fault in the TSC gate-level realization is detectable, every single stuck-at, stuck-open and stuck-on fault in the nMOS network will be detectable.

Case 2: fault in the clocked and inverter transistors.

A stuck-open fault in N1 or P2 is detected by any codeword which makes $z_1=1$ i.e. 0110 or 1001. A stuck-open fault in N2 or P1 is detected by any two-pattern test in which the initialization input is a codeword that makes $z_1=1$, and the test input is a codeword that makes $z_1=0$. Thus, any of the following two-pattern tests can detect N2 or P1 stuck-open: $<0110,0101>$, $<1001,0101>$, $<0110,1010>$, $<1001,1010>$. A stuck-on fault in transistor P1 (P2) is equivalent to a stuck-at 0 (stuck-at 1) fault at z_1, and is hence detectable. A stuck-on fault in transistor N1 or N2 is undetectable. A stuck-at 0 (stuck-at 1) fault at the input of an nMOS transistor is equivalent to a stuck-open (stuck-on) fault, and at the input of a pMOS transistor it is equivalent to a stuck-on (stuck-open) fault. Hence, the previous arguments will also be applicable when stuck-at faults are present at the gates (inputs) of transistors N1, P1, N2 or P2. A stuck-at 0 (stuck-at 1) fault on $\overline{z}_1$ is equivalent to a stuck-at 1 (stuck-at 0) fault at z_1, and is therefore detectable.

Case 3: stuck-at fault on the clock line.

The stuck-at faults at the branches of the clock line are already covered in Case 2. Thus, only the stem of the clock line need be considered. A stuck-at 0 fault on the clock stem is equivalent to the double fault P1 stuck-on and N1 stuck-open. This is equivalent to a stuck-at 1 on $\overline{z}_1$ or a stuck-at 0 at z_1, and is hence detectable. A stuck-at 1 fault on the clock stem is equivalent to a simultaneous presence of P1 stuck-open and N1 stuck-on. This can be detected by any of the two-pattern tests mentioned in Case 2 for detecting P1 stuck-open.

Case 4: undetectable stuck-on fault(s)

Let all the detectable faults mentioned in Cases 1, 2 and 3 belong to set Φ. ϕ will denote any member of Φ. We will use the notation $z_1(f)$ to denote the circuit output in the presence of a single fault or a fault sequence f. Test(f) will denote the set of single or two-patterns tests (codewords) which detect f. For brevity we will denote N_1 (N_2) stuck-on fault as N_1-on (N_2-on). A sequence of faults will be denoted as $<f_1, f_2, .., f_n>$ where the faults f_i, $1 \leq i \leq n$, occur in the sequence shown.

The following can be verified:

(1) $z_1(N_1\text{-on}) = z_1$
(2) $z_1(N_2\text{-on}) = z_1$
(3) $z_1(<N_1-on, N_2-on>) = z_1$
(4) $z_1(<N_2-on, N_1-on>) = z_1$
(5) $\text{Test}(<N_1-on, \phi>) = \text{Test}(\phi)$
(6) $\text{Test}(<N_2-on, \phi>) = \text{Test}(\phi)$
(7) $\text{Test}(<N_1-on, N_2-on, \phi>) = \text{Test}(\phi)$
(8) $\text{Test}(<N_2-on, N_1-on, \phi>) = \text{Test}(\phi)$

From the above we can conclude that the presence of N_1-on or N_2-on or both has no effect on the output z_1. This is true for both input codewords and non-codewords. Furthermore, if a detectable fault ϕ were to occur later then the fault sequence becomes detectable by the test for ϕ. Therefore, we can conclude that the presence of the undetectable stuck-on fault(s) does not affect the logical behavior of the circuit.

The second subcircuit from Fig. 1 can also be implemented using the domino-CMOS technique. This together with the circuit in Fig. 2 will form a domino-CMOS two-variable two-rail checker. The above results will be valid for the implementations of both z_1 and z_2 in this checker.

B. Proof of the Self-Checking Property

We are now in a position to formally establish the self-checking property of the domino-CMOS implementations.

Theorem 1: For a gate-level checker which is TSC with respect to stuck-at faults, the domino-CMOS implementation is SSC with respect to stuck-at, stuck-open and stuck-on faults.

Proof: The strongly self-checking (SSC) property combines the SFS and SCD properties (see Definition 8). Since the TSC gate-level checker, denoted GL, is code-disjoint, its domino-CMOS implementation, denoted DC, is also code-disjoint before the occurrence of any fault.

Suppose that DC has n gate-pairs. Denote the outputs of these gate-pairs as g^i, $1 \leq i \leq n$. The output of the complex gate of the gate-pair g^i will be denoted as $\overline{g}^i$. For simplicity we will also use g^i to denote the gate-pair itself. The clocked and inverter transistors in gate-pair g^i are denoted as $N1^i$,$P1^i$,$N2^i$ and $P2^i$. Every g^i can potentially have the following undetectable faults or fault-sequences: $N1^i$-on, $N2^i$-on, $<N1^i$-on, $N2^i$-on$>$, $<N2^i$-on, $N1^i$-on$>$. Let the set of remaining single faults in the whole circuit be Φ. ϕ denotes any member of Φ. Let ϕ be present in the nMOS network of some gate-pair g^i. Irrespective of whether ϕ is a stuck-at, stuck-open or stuck-on fault, there is a corresponding equivalent stuck-at fault in GL. Hence DC must be self-testing and fault-secure with respect to ϕ. Similar arguments hold if ϕ is $N1^i$-open, $P2^i$-open, $P1^i$-on, $P2^i$-on, $\overline{g}^i$ or g^i stuck-at 0 or stuck-at 1, or clock line feeding g^i stuck-at 0, because all these faults map to stuck-at faults at g^i. If ϕ is $P1^i$-open, $N2^i$-open or clock line stuck-at 1, then a two pattern test made up of two codewords will detect it. Now consider the clock stem which feeds the clock to each (or a subset) of the gate-pairs g^i, $1 \leq i \leq n$. Let ϕ be a stuck-at 0 fault on this clock stem. This is equivalent to a unidirectional stuck-at 0 fault at the lines g^i. This can only cause a unidirectional error at the checker outputs, and is hence detectable. Similarly, a stuck-at 1 fault on the clock stem can be shown to be detectable. Before ϕ is detected, DC remains fault-secure. Thus, for all the above cases, Condition (a) in Definition 8 is met.

Let f^i be one of the following undetectable faults or fault-sequences in g^i: $N1^i$-on, $N2^i$-on, $<N1^i$-on, $N2^i$-on$>$, $<N2^i$-on, $N1^i$-on$>$. From the discussion in Section 4.A it is clear that f^i does not affect the logical behavior of DC. Therefore, DC remains fault-secure and continues to map a non-codeword input to a non-codeword output. Thus Condition (b) of Definition 8 is met. Let ϕ occur after f^i. In other words the fault sequence is $<f^i,\phi>$. We know from the previous discussion that DC behaves in the presence of $<f^i,\phi>$ just the way it would in the presence of ϕ alone. Thus, DC would be fault-secure and self-testing with respect to $<f^i,\phi>$. Another possibility is that a fault f^j occurs after f^i, where f^j is one of the undetectable faults or fault-sequences in gate-pair g^j. In presence of the fault-sequence $<f^i,f^j>$ both g^i and g^j behave normally. Therefore, DC would remain fault-secure and map non-codeword inputs to non-codeword outputs. It is clear that this argument can be extended to any undetectable fault sequence $<f^i,f^j,..,f^m>$. At any point if ϕ occurs then DC would become fault-secure and self-testing. Therefore, DC is SSC.

□

It is worth mentioning at this point that a two-pattern test which detects $N2^i$-open and $P1^i$-open in some gate-pair g^i often also detects other such faults along the sensitized path. Thus only a few two-pattern tests can detect all such stuck-open faults in the circuit.

A theorem similar to Theorem 1 can be stated as follows for functional circuits.

Theorem 2: For a gate-level functional circuit which is TSC with respect to stuck at faults, the domino-CMOS implementation is SFS with respect to stuck-at, stuck-open and stuck-on faults.

Proof: Similar to the proof of Theorem 1.

□

Theorem 2 is geared towards TSC circuits whose outputs are encoded using a unidirectional error detecting code and for which a domino-CMOS implementation is possible.

The above results were derived assuming that in every gate-pair the complex gate as well as the inverter is p-dominant. It is easy to show that the results are valid even if the complex gate is made p-dominant, but the inverter is made n-dominant. However, interested readers are encouraged to verify that if the complex gate is made n-dominant, the proper operation cannot be guaranteed. A CMOS gate can easily be made n-dominant or p-dominant by properly sizing its nMOS and pMOS networks. So this assumption does not put an unreasonable constraint on the designer.

It was pointed out in Section 2.B that the fault-secure property is not required in a checker if its outputs are directly observable. We can prove the following theorem for such checkers.

Theorem 3: For a gate-level checker which is self-testing and code-disjoint with respect to stuck-at faults, the domino-CMOS implementation is SCD with respect to stuck-at, stuck-open and stuck-on faults.

Proof: Similar to the proof of Theorem 1.

□

We saw in Section 3 that a domino-CMOS gate is immune to test set invalidation due to timing-skews. There is another mechanism through which test set invalidation can take place, namely charge-sharing [26]. To protect against this possibility a feedback transistor can be added to every domino-CMOS gate converting it into what is commonly called single-ended cascode voltage switch logic (SCVSL) [29]. Although for simplicity of exposition we derived our results for a domino-CMOS implementation, it is easy to show that they are extensible to SCVSL implementations of TSC circuits.

To implement TSC circuits which contain inverters we need the differential cascode voltage switch logic (DCVSL) [29]. The operation of a DCVSL circuit is similar to the operation of a domino-CMOS or SCVSL circuit with the advantage that both inverting and non-inverting functions can be implemented. DCVSL implementations of TSC circuits are beyond the scope of this paper. However, we have considered such implementations in another paper [30].

5. CONCLUSION

In this paper we showed why a domino-CMOS implementation of a TSC circuit is much superior to its static CMOS implementation. We defined a new property for checkers called the strongly self-checking property. This property is required when the two outputs of the checker are not directly observable. A self-checking system consisting of an SFS functional circuit and an SSC checker can meet the TSC goal even when redundancies are present. The previously defined SCD property [5] is not adequate when the outputs of the checker are not directly observable. We showed how a TSC functional circuit (checker) can be implemented using an SFS (SSC) domino-CMOS circuit based on a comprehensive fault model including stuck-at, stuck-open and stuck-on faults. This fault model can be made even more comprehensive if the layout rules in [19] and [20] are followed.

REFERENCES

[1] Y. Savaria, N.C. Rumin, J.F. Hayes and V.K. Agrawal, "Soft-error filtering: a solution to the reliability problem of future VLSI digital circuits," *IEEE Proceedings,* vol. 74, no. 5, pp. 669-683, May 1986.

[2] W.C. Carter and P.R. Schneider, "Design of dynamically checked computers," in *Proc. IFIP '68,* Edinburgh, Scotland, pp. 878-883, Aug. 1968.

[3] D.A. Anderson and G. Metze, "Design of totally self-checking circuits for m-out-of-n codes," *IEEE Trans. Comput.*, vol. C-22, pp. 263-269, March 1973.

[4] J.E. Smith and G. Metze, "Strongly fault-secure logic networks," *IEEE Trans. Comput.*, vol. C-27, pp. 491-499, June 1978.

[5] M. Nicolaidis, I. Jansch and B. Courtois, "Strongly code-disjoint checkers," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Orlando, FL, pp. 16-21, June 1984.

[6] D.K. Pradhan and J.J. Stiffler, "Error-correcting codes and self-checking circuits in fault-tolerant computers," *IEEE Computer*, pp. 27-37, March 1980.

[7] S.M. Reddy, "A note on self-checking checkers," *IEEE Trans. Comput.*, vol. C-23, pp. 1100-1102, Oct. 1974.

[8] M.A. Marouf and A.D. Friedman, "Efficient design of self-checking checker for any m-out-of-n code," *IEEE Trans. Comput.,* vol. C-27, pp. 482-490, June 1978.

[9] N. Gaitanis and C. Halatsis, "A new design method for m-out-of-n TSC checkers," *IEEE Trans. Comput.*, vol. C-32, pp. 273-283, 1983.

[10] C. Efstathiou and C. Halatsis, "Modular realization of totally self-checking checkers for m-out-of-n codes," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Milan, Italy, pp. 154-161, June 1983.

[11] T. Nanya and Y. Tohma, "A 3-level realization of totally self-checking checkers for m-out-of-n codes," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Milan, Italy, pp. 173-176, June 1983.

[12] S. Piestrak, "Design method of totally self-checking checkers for m-out-of-n codes," in *Proc. Int. Symp. Fault-Tolerant Comput.,* Milan, Italy, pp. 162-168, June 1983.

[13] M.J. Ashajee and S.M. Reddy, "On totally self-checking checkers for separable codes," *IEEE Trans. Comput.*, vol. C-26, pp. 737-744, Aug. 1977.

[14] M.A. Marouf and A.D. Friedman, "Design of self-checking checkers for Berger codes," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Toulouse, France, pp. 179-184, June 1978.

[15] D.A. Anderson, "Design of self-checking digital networks using coding techniques," Tech. Report R-527, CSL, Univ. of Illinois, IL, 1971.

[16] J.E. Smith, "The design of totally self-checking combinational circuits," Tech. Report R-737, CSL, Univ. of Illinois, IL, Aug. 1976.

[17] G.P. Mak, J.A. Abraham and E.S. Davidson, "The design of PLAs with concurrent error detection," in *Proc. Int. Symp. Fault-Tolerant Comput.*, pp. 303-310, June 1982.

[18] N.K. Jha and J.A. Abraham, "Totally self-checking MOS circuits under realistic physical failures," in *Proc. Int. Conf. Comp. Design,* Port Chester, NY, October, 1984.

[19] M. Nicolaidis and B. Courtois, "Layout rules for the design of self-checking circuits," in *Proc. VLSI Conf.*, Tokyo, August, 1985.

[20] N.K. Jha and J.A. Abraham, "Techniques for efficiently implementing totally self-checking checkers in MOS technology," *Int. Journal of Comp.& Math with applications,* vol. 13, No. 516, pp. 555-566, 1987.

[21] A.M. Paschalis, D. Nikolos and C. Halatsis, "Efficient modular design of TSC checkers for m-out-of-n codes," *IEEE Trans. Comput.*, vol. 37, pp. 301-309, March 1988.

[22] J. Galiay, Y. Crouzet and M. Vergniault, "Physical versus logical fault models for MOS

LSI circuits: impact on their testability," *IEEE Trans. Comput.*, vol. C-29, pp. 527-531, June 1980.

[23] R.L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits," *Bell Syst. Tech. Journal*, vol. 57, pp. 1449-1474, May-June 1978.

[24] S.M. Reddy, M.K. Reddy and J.G. Kuhl, "On testable design for CMOS logic circuits," in *Proc. Int. Test Conf.*, Philadelphia, PA, pp. 435-445, October, 1983.

[25] N.K. Jha and J.A. Abraham, "Design of testable CMOS logic circuits under arbitrary delays," *IEEE Trans. on CAD of ICAS*, vol. CAD-4, No. 3, pp. 264-269, July 1985.

[26] S.M. Reddy and M.K. Reddy, "Testable realization for FET stuck-open faults in CMOS combinational logic circuits," *IEEE Trans. Comput.*, pp. 742-754, August 1986.

[27] S. Kundu and S.M. Reddy, "On the design of robust testable CMOS combinational logic circuit," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Tokyo, June 1988.

[28] S. Kundu, S.M. Reddy and N.K. Jha, "On the design of robust multiple fault testable CMOS combinational logic circuits," to appear in *Int. Conf. on CAD*, Santa Clara, CA, Nov. 1988.

[29] L.G. Heller, W.R. Griffin, J.W. Davis and N.G. Thoma, "Cascode voltage switch logic: a differential CMOS logic family," in *Proc. Int. Solid-State Circuits Conf.*, pp. 16-17, Feb. 1984.

[30] N.K. Jha, "Testing of cascode voltage switch parity trees: application to parity and two-rail checkers," submitted to *IEEE Trans. on CAD of ICAS*.

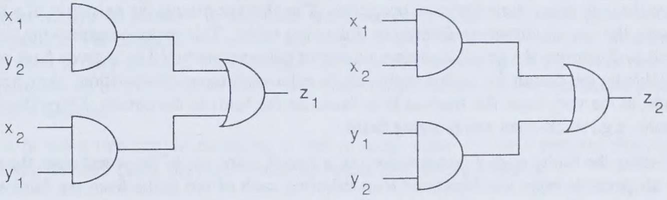


Fig. 1. Gate-level implementation of TRC_2

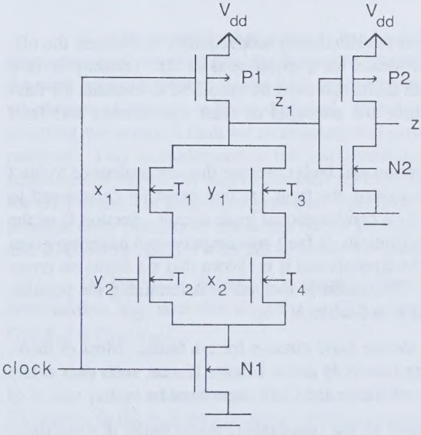


Fig. 2. Domino-CMOS implementation of a subcircuit of TRC_2

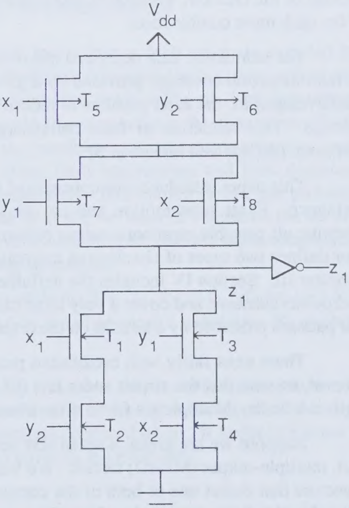


Fig. 3. Static CMOS implementation of a subcircuit of TRC_2

FAULT COLLAPSING FOR CONCURRENT ERROR DETECTION ANALYSIS

L. L. KINNEY and T. RAMAKRISHNAN
Department of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

ABSTRACT

The concepts of fault subsumption and covering are introduced. Error checkpoints which determine most subsumption and covering relationships are defined and they reduce the fault set required to evaluate the fault coverage of concurrent error detection. Concurrent detection of double faults using single fault test sets is discussed.

I. INTRODUCTION

Concurrent error detection schemes typically have a monitor or checker connected to the circuit output lines and, possibly, some of the circuit input lines [ref. 2,3,6]. The circuit is designed so a set of faults that are likely to occur in the system produce erroneous output combinations that differ from any output combination that can occur during normal operation. The checker detects the existence of a fault in the circuit by monitoring the circuit outputs to determine if they are valid. This paper considers the task of analyzing a given circuit to determine the possible erroneous output patterns produced by a given fault set. Note that this requires analyzing the circuit for each possible fault under each input combination. Any acceptable fault set must include, at the very least, the stuck-at (s-a) faults on the leads in the circuit. Often the set includes other types of faults, e.g., stuck-open and bridging faults.

Finding the faulty output combinations of a circuit involves, in the worst case, the simulation of the circuit for all possible input combinations after injecting each of the faults from the fault set one at a time. This is computationally expensive for any but the simplest circuits. However, in order to evaluate the fault coverage of the checker, we need to determine the circuit's output combination for each of the faults in the list and for each input combination.

The simulation task described above is analogous to the simulation task required to evaluate the off-line fault detection coverage provided by a given set of test vectors for a specified fault set. Techniques have been developed for the latter problem to reduce the number of faults that must be simulated to evaluate the fault coverage. This reduction or fault collapsing approach uses the concepts of fault equivalence and fault dominance that are well known [1,5].

This paper introduces the concepts of fault subsumption and fault covering that are analogous to fault dominance. Fault subsumption and covering reduce the size of the fault set that must be considered to determine all possible erroneous output patterns produced by a combinational logic circuit. Section II of the paper defines two types of checkers as motivation for the definitions of fault subsumption and covering given in section III. Section IV includes the definition of error-checkpoints and it is shown that s-a faults on error-checkpoints subsume and cover a very large class of faults. Discussion of the task of determining the possible error patterns produced by s-a faults on the error-checkpoints is in Section V.

There exist fairly well established procedures for testing logic circuits for s-a faults. Most of them, however, assume that the circuit under test (CUT) has at any time only one s-a fault. Hence, such tests detect single, s-a faults. Multiple s-a faults arise when more than one line in the CUT experiences a fault.

Suppose we are given a set of test vectors that detect all the (detectable) single faults in a multiple-input, multiple-output (MIMO) circuit. We would like to find out how many of the double faults are detected by vectors that detect one or both of the component single faults. Detection of double faults (and if possible faults of multiplicity greater than 2) in this manner is of interest when we consider concurrent fault detection based on the use of keys [3]. This is discussed in section VI.

II. CLASSIFICATION OF ERROR CHECKERS

We classify checkers into two types called linear and nonlinear. In both cases, the circuit outputs are encoded. Thus, some output combinations are "legal" and are generated during normal operation while other combinations are "illegal", i.e., they can only occur if there is a fault in the circuit. The design of such circuits requires specification of a fault set and derivation of any errors which they can generate. Based upon the possible errors, the effectiveness of the encoding of the outputs is evaluated. In order to discuss the two types of checkers, the concepts of error patterns and erroneous output patterns produced by a fault are needed.

In the following definitions **G** represents a vector of output values for a correctly functioning, multiple output circuit, $\mathbf{G_{fp}}$ represents a vector of output values for the circuit containing fault f_p and X, X_m, etc. represent circuit input combinations.

Definition 1: The set of erroneous output patterns (EOP) produced by a fault f_p is $EOP(f_p) = \{G_{fp}(X_m) \mid G_{fp}(X_m) \neq G(X_m)$ for all input combinations $X_m\}$.

Definition 2: The set of error patterns (EP) produced by a fault f_p is $EP(f_p) = \{G_{fp}(X_m)$ XOR $G(X_m) \mid G_{fp}(X_m) \neq G(X_m)$ for all input combinations $X_m\}$.

In a linear checker, the error patterns produced by faults determine if the faults are detected and the detectability is independent of the specific erroneous output pattern produced. An example is the duplication of a circuit as illustrated in Figure 1. The checker is simply an exclusive-or (comparator) and it detects any fault that generates the error patterns 01 or 10. The detection of these error patterns is independent of the correct output from the circuit.

Figure 2 shows a simple example of a nonlinear checker. The normal circuit output patterns are 10, 11, and 01; the pattern 00 is only produced if there is a fault in the circuit. The checker signals an error when the erroneous output pattern 00 occurs. In this case, the detection of a fault depends upon more than the associated output pattern. For example, a s-a-0 on the output of the NAND gate can only produce the error pattern 10 but this fault is only detected if the correct output pattern should be 10 and is not detected if the correct output pattern should be 11. In other words, this fault can produce the erroneous output patterns 00 or 01 but only the first is detected.

The point of these two simple examples is that in some cases the error patterns determine if the checker detects a fault. In other cases, the exact erroneous output patterns are required.

III. DEFINITION OF FAULT RELATIONSHIPS

In order to verify that a circuit has been designed so all faults from a fault set S are detected by a concurrent error checker, the circuit must be simulated for each fault and each input combination, to find either the error pattern produced by the fault or the erroneous output combination produced by the fault depending on which type of checker is being designed. In either case, we would like to minimize the simulation effort by selecting the smallest fault set to simulate that produces all of the possible error patterns or erroneous output patterns. This is analogous to the test generation problem where fault equivalence and fault dominance relationships reduce the size of the fault set examined for test generation. To a certain extent, our problem is the dual of the test generation problem. Specifically, for a given fault set, we want to determine the set of all possible error patterns or erroneous output patterns that can occur considering all possible input combinations and all possible faults. With this in mind, we make the following definitions:

Definition 3: A fault f_p is (weakly) covered by the set of faults $S = \{f_1, f_2, \ldots, f_k\}$ if, for each input combination X_m, such that $G_{fp}(X_m) \neq G(X_m)$ there is a fault f_j in S and an input combination X_n such that $G_{fj}(X_n) = G_{fp}(X_m)$.

In other words, a set of faults, S, cover a fault, fp, if every erroneous output combination produced by fp is also produced by at least one of the faults in S (but not necessarily under the same input combination). Of course, faults in S may produce erroneous output combinations not produced by fp. If faults in S cover fp, then it is not necessary to simulate fp to determine all EOPs. In terms of the EOP sets,

$$EOP_{fp} \leq U_j \{EOP_{fj} \mid \text{for all } f_j \text{ in } S\}.$$

Coverage satisfies transitivity. For example, if R, S, and T are three sets of faults such that each fault in R is covered by S and each fault in S is covered by T, then each fault in R is covered by T.

Definition 4: A fault f_p is strongly covered by the set of faults $S = \{f_1, f_2,...,f_k\}$ if, for each input combination X_m such that $G_{fp}(X_m) \neq G(X_m)$, there is a fault f_j in S such that $G_{fj}(X_m) = G_{fp}(X_m)$.

Strong covering requires that the covering fault produce the same erroneous pattern for the same input combination.

Definition 5: A fault f_p is subsumed by the set of faults $S = \{f_1, f_2,..., f_k\}$ if for each input combination X_m such that $G_{fp}(X_m) \neq G(X)$ there is a fault f_j and an input combination X_n such that $G_{fj}(X_n)$ XOR $G(X_n) = G_{fp}(X_m)$ XOR $G(X_m)$

A fault, f_p, is subsumed by a set of faults, S, if every error pattern that can be caused by f_p is also caused by one of the faults in S. Of course, there may be many error patterns caused by faults in S different from those caused by f_p. In terms of the error sets, $E_{fp} \leq U_j \{E_{fj} \mid \text{for all } f_j \text{ in } S\}$.

Subsumption satisfies transitivity. For example, if R, S, and T are three sets of faults such that each fault in R is subsumed by S and each fault in S is subsumed by T, then each fault in R is subsumed by T.

IV. ERROR-CHECKPOINTS

We would like an easy procedure for determining fault sets which subsume (cover) a given fault and, in particular, an easy procedure for determining a 'minimal' set of faults that subsumes (covers) all the faults in a given fault set. For example, let F be the set of all single, s-a faults that can exist in a combinational circuit. What is a small set, S, of faults such that each fault in F is subsumed (covered) by the faults in S?

Finding the minimal set S is a difficult problem but it is relatively easy to find an S much smaller than the set F. First, we relate the concept of dominance in test generation to the concept of subsumption. If fault f_j dominates fault f_p, then for purposes of determining the error set for the two faults, the fault f_p can be ignored if the error patterns produced by f_p are also produced by f_j. The following definition and lemma specify one situation of dominance where this condition would be satisfied.

Definition 6: Given a circuit implementation and a lead, r, and outputs, $g_1, g_2,..., g_n$, then a lead s (gate h) is a separating lead (separating gate) for lead r if and only if all paths from r to any output pass through s (h).

We write r s-a-v, or equivalently, r/v to denote the fact that line r is (erroneously) stuck at v where v can have the value 0 or 1.

Lemma 1: If lead s is a separating lead for lead r, then the set {s/0,s/1} subsumes (covers) the fault r/0 and, also, the fault r/1. (This includes cases where there is more than one path from r to s, possibly with differing inversion parities.)

Proof: Consider the fault r/0 and any input combination. If r/0 causes an erroneous output, then s either has an incorrect value of 0 or an incorrect value of 1. However, s/0 or s/1 would cause the same erroneous outputs since error effects from r must pass through s.

Lemma 2: If lead s is a separating lead for lead r and every path from r to s has even (odd) parity, then the fault s/a (s/a') subsumes and, also, covers the fault r/a for a = 0 and 1.

As a special case, assume that s is a separating lead for lead r and that lead s only has paths to a single output, say g_j. Further, assume that either fault r/0 or fault r/1 is detectable. Since one of these two faults is detectable, there is an input combination which causes an error pattern with only g_j in error. However, this is the only error pattern generated by a fault at s; thus, s/0 and s/1 are both subsumed by r/0, if it is detectable, and by r/1, if it is detectable. We have proved the following lemma.

Lemma 3: Let lead s be a separating lead for lead r and such that every path from s is to a unique output. Then, if r/a, a=0 or 1, is detectable, r/a subsumes s/0. Also, r/a subsumes s/1.

Thus, when the lead r and its separating lead s only have paths to a single output, the subsumption relationship applies in both directions. However, the preceding lemma is not true for fault covering. A fault at s may generate an erroneous output pattern for some input combination that does not detect a fault at r and, consequently, there can be more erroneous output patterns for faults at s than for faults at r.

The preceding discussion of separating leads can be generalized somewhat. Consider a cone of logic within a circuit that has a single output h. [Figure 3] Note that the only restriction on the cone is that it have a single output; the cone may have any type of structure including reconvergent fanout. Consider a fault f in the cone, not necessarily a s-a fault, either single or multiple, which causes an error at the circuit output for some input combination $\mathbf{X}_m$. Then, the output of the cone is erroneous for $\mathbf{X}_m$, say it is incorrectly equal to v. However, the single, s-a fault h/v' also produces the same error at the circuit outputs for $\mathbf{X}_m$. Clearly, the two faults h/0 and h/1 subsume (cover) any faults that can exist in the cone.

Lemma 4: Let F be any fault set that only affects a cone of logic with single output, h, in a combinational logic circuit. Then, F is subsumed (and also strongly covered) by the fault set S = {h/0,h/1}.

(Note that F can contain faults of any nature which only affect the cone, for example, faults which cause sequential behavior within the cone are permissible. However, F cannot include faults which affect more than one cone at the same time).

Definition 7: Let $\{c_1, c_2, ..., c_r\}$ be a collection of single output cones that partition the logic of a combinational circuit C, i.e., each gate of C is in one, and only one, of the cones. A set of error-checkpoints, EC, for C is the set of output leads of the cones.

The error-checkpoints of a circuit are not unique. For example, one simple set of error-checkpoints is the collection of all fanout stems in the circuit plus the external outputs of the circuit. Note the dual nature of this particular set of error-checkpoints with respect to the earlier definition of checkpoints for use with fault dominance. The latter type of checkpoints is the set of all fanout branches in the circuit plus the external inputs to the circuit.

The preceding definition of error-checkpoints and lemma 4 imply the following result.

Theorem 1: Consider a combinational circuit C, redundant or irredundant. Let $\{c_1, c_2, ..., c_r\}$ be a collection of single output cones that partition the logic of C and let EC be the error-checkpoints specified by the cones.

(i) Let F be any set of faults that individually cause errors within a single cone and let S be the set of single, s-a faults on the EC points within C. Then, S subsumes F and S also strongly covers F.

(ii) Let F be any set of faults that cause errors within one or more cones and let S be the set of multiple, s-a faults on the EC points within C. Then, S subsumes F and S also strongly covers F.

CMOS Stuck-Open and MOS Faults

This result holds for stuck-open faults in CMOS and can be extended to arbitrary faults, including those that cause sequential behavior. This will include many of the short and open faults that can occur in MOS logic. These faults can cause major changes in the logic function of complex gates and they do not have simple models such as the s-a model.

Lemma 5: Let F be the set of all single, s-a faults and single, stuck-open faults within gates of a CMOS combinational logic circuit and let S be the set of single, s-a faults at a set of error-checkpoints within the circuit. Then, the error sets of F and S are equal.

Bridging Faults

Consider bridging faults within a network. Let cones 1 and 2 be any two cones within the network that each have a single output. Any bridging fault contained within one of the cones will be subsumed by a single, s-a fault on the cone output. Next, consider a bridging fault between a line in cone 1 and a line in cone 2. If the two lines have the same signal value, then no error results. If the two lines have opposite values and if the bridge causes an AND effect, then the line which should have a 1 is in error but the other line is not in error; hence, only one cone has an erroneous output. Similarly, if the two lines have opposite values and if the bridge causes an OR effect, then the line which should have a 0 is in error but the other line is not in error; hence, again only one cone is affected. Therefore, in any case, a bridging fault between two lines in a circuit will be subsumed by the single, s-a faults on the outputs of single output cones within a circuit.

V. DETERMINATION OF ERROR PATTERNS FOR A FAULT ON A FANOUT STEM

Let PI and PO stand for Primary Input and Primary Output, respectively.

There is a structural span (s_span) for each lead in a circuit that is based simply on the paths of the circuit. The s_span of lead k is the set of circuit outputs for which a path exists from k to each of the outputs. There is also a functional span (f_span) for a lead k which is determined by the input combination and the functional dependence of the outputs. For a particular input combination, the f_span of k is the set of circuit outputs which would change if the signal value on k changed, i.e., it is the set of circuit outputs sensitive to the value on k. The cardinality of the f_span set has a minimal and maximal value determined by considering all possible input combinations. Obviously, the f_span set must satisfy the following relationships:

$$\mathbf{0} \leq \text{f\_span}(k,\mathbf{X}) \leq \text{s\_span}(k) \leq \mathbf{I}, \text{ and}$$

$$0 \leq |\text{f\_span}(k,\mathbf{X})| \leq |\text{s\_span}(k)| \leq p,$$

where **0** is the null set, **I** is the set of all POs, **X** is any input vector, and $p = |\mathbf{I}|$ is the number of POs in the circuit.

If lead k is a fanout-stem and if leads $r_1, r_2, \ldots, r_m$ are its fanout branches, the s_span of k is the union of the s_spans of its fanout branches; thus,

$$\text{s\_span}(k) = \text{U}_j \; \{\text{s\_span}(r_j) \mid \text{for all } r_j\}, \text{ and}$$

$$|\text{s\_span}(k)| \leq \text{SUM}_j \; |\{\text{s\_span}(r_j) \mid \text{for all } r_j\}|.$$

Similar relationships apply to the f_spans for each input combination.

The relationship between functional and structural span is not easily determined for a particular circuit and a particular input combination. The following examples illustrate some of the various relationships that may exist between f_span and s_span.

Lemma 6: There exist circuits satisfying the following relationships for some of the circuits' input lines:

a) $|\text{f\_span}|$ = a constant, i.e., the cardinality of the f_span is independent of the input combination. (A Demultiplexer is an example.)

b) $|\text{f\_span}| = |\text{s\_span}|$ for every input combination. (An example is an XOR (parity) tree.)

c) $|\text{f\_span}| < |\text{s\_span}|$. (Examples are decoders and demultiplexers.)

Example 1 lists the f_span and s_span sets for various lines in a 4-to-1 Demultiplexer. Note that the data input has a structural span of cardinality 4, but its functional span has cardinality 1 independent of the input combination. The control inputs have functional spans of cardinality 0 or 2 and structural spans of cardinality 4.

Additional structural and functional span information can be given using matrices similar to the matrices used to derive verification tests [4]. In these matrices, the rows correspond to leads within the circuit, e.g., error-checkpoints, and the columns correspond to circuit outputs. In the structural span matrix, the (i,j) entry is 0 if there is no path from i to j, 1 if there are only paths of even inversion parity from i to j, -1 if there are only paths of odd inversion parity from i to j, and 2 if there are paths of both even and odd inversion parity from i to j. Functional span matrices are similarly defined but the entries indicate the existence of a path and that the path is sensitive to the value on the input lead; therefore, functional span matrices are dependent on the input combination. Example 2 illustrates the matrix description.

The minimum and maximum f_spans have particular significance when the outputs of a circuit are encoded using a linear code. If the maximum f_span of any line in the circuit is less than the minimum distance of the code, then the encoding is sufficient to permit a checker to concurrently detect all errors caused by any single, s-a fault within the circuit and, from Theorem 1, the errors caused by many other faults. On the other hand, if every line within the circuit has a minimum f-span that is less than the minimum distance of the code, then the encoding permits detecting some of the errors caused by any single, s-a fault in the circuit. In this case, not every fault is concurrently detected but each fault is eventually detected assuming that all input combinations are eventually applied to the circuit.

VI. DETECTION OF DOUBLE FAULTS USING SINGLE FAULT TEST VECTORS

The preceding sections were concerned with the design of a circuit that would concurrently detect a set of faults in a given circuit, primarily the set of all single, s-a faults. Once such a concurrent error detector has been designed for a specific set of faults, it may be of interest to know how effective it is in detecting a more general class of faults. Specifically, in this section we assume that the concurrent error detector has been designed to detect all single, s-a faults, and we want to determine how effective it is in concurrently detecting double, s-a faults, i.e., s-a faults with two components. If a double fault is concurrently detected by the error detector, all error patterns generated by the double fault would have to be detected, i.e., each error pattern generated by the double fault would be the same as an error pattern generated by some single fault for some input combination. (This assumes that the concurrent error detector detects exactly the error patterns caused by single faults and no others.)

The determination of double faults that satisfy this condition appears difficult. Futhermore, it appears unlikely that many double faults would satisfy this condition in any circuit. Therefore, the effectiveness of the error detector for double faults must be evaluated differently. Instead of determining which double faults only cause error patterns that equal error patterns caused by single faults, we try to determine which double faults cause some error patterns that are also caused by single faults. For some input combinations, such a double fault would cause an error pattern different from all the error patterns caused by single faults and, therefore, the double fault would be undetected for that input combination. On the other hand, there would be some other input combination for which the double fault generates an error pattern that is included in the set of error patterns caused by single faults and, therefore, the double fault would be detected for that input combination.

We want to identify those double faults that are detected part of the time by the single fault, concurrent error detector. If all input combinations are applied to the circuit during its operation, such double faults would eventually be detected. This could also be viewed as finding the double faults that are detected with some latency, as opposed to concurrent detection, where the latency would be determined by how frequently all input combinations are applied to the circuit.

To set the stage for the following discussion, we mention one other view of the problem we are trying to solve. We are trying to identify those double faults that are detected by an input combination, a test vector, which detects some single fault with an additional restriction. Not only must the double fault be detected by the same single fault test vector but the double fault must cause exactly the same error pattern as the one caused by the single fault. The latter restriction is necessary for the single fault, concurrent error detector to detect the double fault.

The following discussion denotes the double fault as (r/v,s/w) where r and s are the two lines on which the two components of the double fault reside. A completely general version of our problem would be "Is there a single fault q/u and two input vectors a and b such that the error pattern generated by the circuit with fault (r/v,s/w) under input a is the same as the error pattern generated by the circuit with fault q/u under input b?". However, the following results apply to the restricted case where q/u is the same as r/v or s/w and where a is the same as b.

Preliminaries

We assign every pair of lines in the CUT into one of four classes depending on their s_span. We then list the conditions under which we can guarantee the detection of double faults in varying measures of success for each of the four classes.

Definition 8: Let r be any line in the circuit. Then,
PI[r] = {p|p is a PI that feeds r}, and
PO[r] = {p|p is a PO fed by r} define the PIset and the POset of line r.
Note that PO[r] is the s_span of r.

Definition 9: T[f] = {t|t is a test vector for fault f} is the set of test vectors for fault f in a circuit.

Suppose an input vector t detects a fault f (of multiplicity ≥ 1). The presence of f would cause one or more of the POs to assume values that would be complements of their normal values for t.

Consider two lines r and s in a circuit. There may be some PIs that feed both r and s. In addition, there may be one or more POs fed by both r and s. Thus, there are four possibilities to consider. Based on these, we have the following definition.

Definition 10:

A pair of lines (r,s) is in class	If the set PI[r].PI[s] is	PO[r].PO[s]
P1	Null	Null
P2	Null	Not null
P3	Not null	Null
P4	Not null	Not null

Let us denote the erroneous output pattern due to fault f on test t by EOP(f,t).

Definition 11: Suppose t is a test for a single fault a. We say that t s_detects the double fault d=(a, b) provided EOP(a,t) = EOP(d,t). Equivalently, we say that d is a-detected, if this holds.

Similarly, if a test u for b is also a test for d in the sense that EOP(b,u) = EOP(d,u), we say that d is b-detected.

All of the discussion that follows applies equally well if s and w are used in place of r and v. Hence, without loss of generality, we list most conclusions only with respect to r/v.

S Detection in Pairs in Classes P1 and P2

Lemma 7: If (r,s) is in P1 or P2, (r/v,s/w) is both r/v_ as well as s/w_detected.

Proof: Since r and s have no common PI, we can set s(t)=w for any t ε T[r/v]. EOP(r/v,t) will then be the same as EOP((r/v,s/w),t).

S Detection in Pairs in Classes P3 and P4

Lemma 8: If (r,s) is in P3 or P4, a vector t ε T[r/v] s_detects (r/v,s/w) provided s(t)=w.

Proof: Since s(t)=w, the presence of the fault s/w does not affect the detection of r/v by t. Hence, EOP(r/v,t) = EOP((r/v,s/w),t). Therefore, t s_detects the double fault; equivalently, the double fault is r/v_detected.

In a test t for r/v, it would be easier to set s to a desired value (either w or w') if (r,s) is a P3 pair than if it were a P4 pair. To see why this is the case, let us consider the circuit in example 3. It shows tests for the single, s-a fault r/v, v ε {0,1}. Let t be one such test. Note that s(t)=0 for each t listed in that example. Thus, the double fault (r/v,s/0) is s_detected by t. One way that t might s_detect (r/v,s/1) is to set s(t)=1. But in this example, PO 'A', at which r/v is detected by t, would not be sensitive to the value on r!

Thus, one possibility for s_detection of a double fault (by a test for a single fault on one of the two lines) arises when the second line has the same value w (both when it is fault-free and also during the double fault) [example 3, fault (r/v,s/0)]. s_detection is also possible when the second line has the complement of that value (w') [example 4].

Theorem 2: In both P3 and P4 pairs, at least 50% of the double faults are s_detected.

Proof: Consider the double faults d1 = (r/v,s/w) and d2 = (r/v,s/w'). Let

T1 = {t|t is a member of T[r/v] and s(t)=w} and

T2 = {t|t is a member of T[r/v] and s(t)=w'}.

Obviously, T = T1+T2 and T1.T2 is null. Since we are considering only detectable faults, T cannot be null for any fault f. If neither T1 nor T2 is null, then both d1 and d2 are s_detected by vectors in T. If T1 is null, d2 is s_detected by T. If T2 is null, d1 is s_detected by T. Thus, at least one of the four possible double faults involving two lines is s_detected by vectors that detect a single fault on one of them. From symmetry, it is clear that the vectors in T[s/w] detect at least one of the remaining double faults.

One of our goals is to identify pairs of lines amenable to s_detection. We start by finding the minimum requirements on the two lines to enable us to set them to any value independent of each other. Using fault subsumption and/or covering, we can achieve a notable reduction in the number of cases to be considered. We are working on identifying circuit configurations that permit s_detection in more than 50% of the double faults.

VII. CONCLUSIONS

The concepts of fault subsumption and fault covering introduced in this paper are useful when analyzing the capabilities of a concurrent error detection monitor. They help reduce the size of the fault set that is considered when determining the fault detection coverage of the monitor. It is easy to identify many of the fault subsumption and fault covering relationships from the structure of the circuit. For example, the error-checkpoints of the circuit are the outputs of single output cones of logic in the circuit and s-a faults on the error-checkpoints subsume and cover most other faults within the circuit. Therefore, error-checkpoints reduce the fault simulation task.

Current work is further developing these concepts to determine if additional properties of a monitor can be easily derived. For example, the functional and structural span matrices defined in this paper determine the error patterns for a fault and, hence, the fault coverage of a monitor.

The use of fault covering and subsumption, coupled with the concept of monitors should be a valuable aid in designing circuits with specified single and double fault coverage parameters. Hence, in addition to studying the single fault coverage of monitors, we are exploring ways of determining their double fault coverage.

VIII. REFERENCES

1 Breuer, M. A., and A. D. Friedman, Diagnosis and Reliable Design of Digital Systems, Computer Science Press, Woodland Hills, Calif., 1976.

2 Guha, A. and L. L. Kinney, "Monitors of Varying Complexity for Concurrently Testing Microprogrammed Control Units," 1984 Internat'l Conf. on Computer-Aided Design, Nov. 12-15, 1984, pp. 137-139.

3 Iyengar, Vijay S. and L. L. Kinney, "Concurrent Fault Detection in Microprogrammed Control Units," IEEE Trans. on Computers, Sept. 1985, pp. 810-821.

4 McCluskey, E. J., "Built-in Verification Test", Proc. IEEE Test Conf., pp. 183-190, 1982.

5 Pradhan, D. K., (ed) Fault Tolerant Computing Theory and Techniques, Vol I, Prentice-Hall, New Jersey, 1986.

6 Wakerly, J. F., Error Detecting Codes, Self-Checking Circuits and Applications, Elsevier North-Holland, New York, 1978.

FIGURES

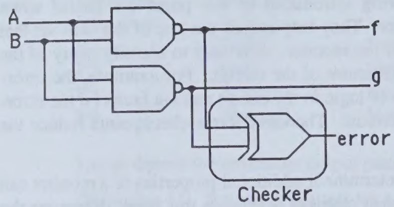
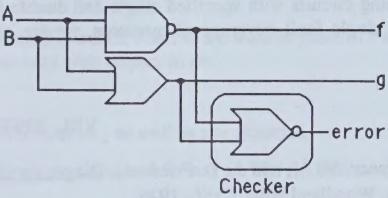

A
B
f
g
error
Checker

Figure 1. Example of a linear checker.


A
B
f
g
error
Checker

inputs		Normal circuit			Faulty circuit (NAND s/0)			
A	B	f	g	checker	f	g	error pattern	checker
0	0	1	0	0	0	0	10	1
0	1	1	1	0	0	1	10	0
1	0	1	1	0	0	1	10	0
1	1	0	1	0	0	1	00	0

Figure 2. Example of a nonlinear checker.

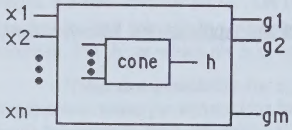

x1
x2
xn
cone
h
g1
g2
gm

Figure 3. Illustration of a cone.

$f_0 = A'B'd$

$f_1 = A'Bd$

$f_2 = AB'd$

$f_3 = ABd$

$s\_span(d) = 4$

$s\_span(A) = 4$

$s\_span(B) = 4$

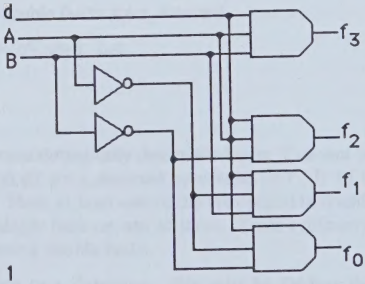
$|f\_span(d)| = 1$

$|f\_span(A,--0)| = 0$

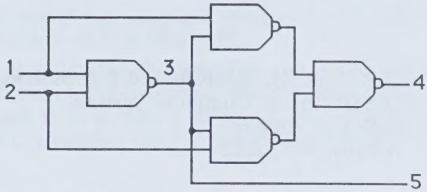
$|f\_span(A,--1)| = 2$

$|f\_span(B,--0)| = 0$

$|f\_span(B,--1)| = 2$


d
A
B
f_3
f_2
f_1
f_0

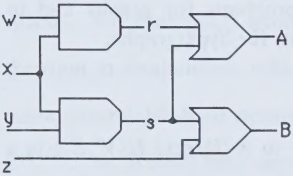
Example 1: 4-to-1 Demultiplexer.



	s_span		f_span							
	4	5	4	5	4	5	4	5	4	5
1	2	-1	1	0	-1	-1	1	-1	-1	-1
2	2	-1	1	0	1	-1	-1	-1	-1	-1
3	1	1	-1	1	1	1	1	1	1	1
(1,2):			00		01		10		11	

Example 2: Structural and Functional Span matrices.

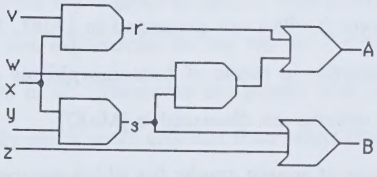
(r,s) is a P3 pair.



T[r/0] = {(110x)}
T[r/1] = {(x0xx),(0x0x)}
Let t be any vector in either of these sets of test vectors.
s(t)=0.
Hence, t s_detects (r/v,s/0) for v=0 or 1.
But, no test for r/v s_detects s/1 because s=1 masks the fault effect on r from reaching A, the only PO in PO[r].

Example 3.

(r,s) is a P3 pair.



T[r/0] = {(110xx)}
T[r/1] = {(0x0xx),(x00xx)}
Let t be any vector in T[r/0].
It s_detects (r/0,s/x).

Similarly, any vector in T[r/1] s_detects (r/1,s/x).

Example 4

SEPARATORS, GRAPH HOMOMORPHISMS AND CHROMATIC POLYNOMIALS[†]

H.B.Hunt III, S.S.Ravi and R.E.Stearns
Department of Computer Science
SUNY at Albany
Albany, NY 12222

ABSTRACT

The problem of determining the existence of a homomorphism from a (source) graph G to a (target) graph H is, in general, NP-complete. We consider this problem for several classes of source graphs including planar graphs, graphs of constant genus, near-planar graphs and bandwidth-limited graphs. Using appropriate separator theorems and the technique of nonserial dynamic programming, we develop algorithms for testing the existence of a homomorphism, finding a homomorphism if one exists and counting the number of homomorphisms. These algorithms are in turn used to develop algorithms for computing the chromatic polynomials of the graphs. Our algorithms are exponential in the *size of the separator* rather than the number of nodes of the graph and hence, are more efficient for graphs with small separators. Our techniques are also applicable to edge coloring problems for graphs and to homomorphism, node coloring and edge coloring problems for hypergraphs.

1. INTRODUCTION

A **homomorphism** from a (source) graph $G(V,E)$ to a (target) $H(V',E')$ is a function $f: V \rightarrow V'$ such that for every edge $\{x,y\} \in E$, the edge $\{f(x),f(y)\} \in E'$. It is easily seen that homomorphisms generalize the notion of vertex coloring [Al87, MS81]. For example, a graph G is r-colorable if and only if there is a homomorphism from G to the complete graph on r nodes (denoted by K_r). Since the graph r-colorability problem is NP-complete for every *fixed* $r \geq 3$ [GJ79], determining whether there is a homomorphism from a graph G to a graph H is NP-complete even for *fixed* targets. Results for specific target families are presented in [Al87, MS81]. Characterizations of several graph properties in terms of homomorphisms are discussed in [FH86]. Applications to secure systems are discussed in [Mo87].

Here we study several classes of source graphs for which appropriate "separator theorems" hold. These are defined as follows:

[†] Research supported in part by NSF Grants No. DCR86-03184, DCI86-03318 and DCR83-03932 respectively.

Definition 1.1:

(i) A class **C** of graphs is **closed** under the subgraph relation if and only if G in **C** implies that every induced subgraph of G is also in **C**.

(ii) A class **C** of graphs which is closed under the subgraph relation is said to satisfy an $f(n)$*–separator theorem* if there exist constants p and q, with $0 < p < 1$ and $q > 0$, such that for every graph G in **C** with n nodes, the node set of G can be partitioned into sets A, B and C satisfying the following three conditions:

   (a) $|C| \leq q\, f(n)$,
   (b) $|A| \leq pn$, $|B| \leq pn$, and
   (c) No edge of G joins a node in A with a node in B.

Separator theorems have been established for several classes of graphs. For example, planar graphs satisfy an $O(\sqrt{n})$-separator theorem [LT79]. Graphs of genus g satisfy a $O(\sqrt{gn})$-separator theorem [GH84] and series-parallel graphs satisfy a 2-separator theorem [HT86]. We will see later in this paper that graphs of bandwidth b satisfy a b-separator theorem.

Separator theorems are very useful in devising efficient divide-and-conquer algorithms. For example, Lipton and Tarjan [LT80] have shown that the planar separator theorem in conjunction with nonserial dynamic programming [Ro82] can be used to solve several NP-hard optimization problems on planar graphs with n nodes in time $2^{O(\sqrt{n})}$. In [RH87] their approach was extended to compute the number of solutions for several such problems in time $2^{O(\sqrt{n})}$. In this paper, we show that a combination of separator and dynamic programming techniques can be used uniformly to solve the following homomorphism problems:

Given a graph G (source) with n nodes and a graph H (target) with m nodes,

(i) Determine whether there is a homomorphism from G to H,
(ii) Find a homomorphism from G to H, if one exists, and
(iii) Find the number of distinct homomorphisms from G to H.

Our algorithms are exponential in the *size of the separator* for G rather than the number of nodes in G. Therefore for graphs with small separators, our algorithms are asymptotically much more efficient than exhaustive search.

To illustrate, consider the case when the source graph G is planar. Since graph 3-colorability is NP-hard for planar graphs [GJ79], it follows that the above problems are NP-hard for planar source graphs even when H is *fixed*. Our algorithms run in time $2^{O(\sqrt{n})}$ when H is fixed and in time $m^{O(\sqrt{n})}$ (equivalently, in time $2^{O(\sqrt{n}\log m)}$)

when H is a part of the problem instance and has m nodes. A consequence of our results is that the **chromatic polynomial** of a planar graph with n nodes can be computed in time $n^{O(\sqrt{n})}$.

Our results extend to any class of source graphs G which have small separators. Thus chromatic polynomials for graphs with separators of a constant size can be computed in polynomial time. Graphs in this class include trees[1], series-parallel graphs and graphs of constant bandwidth. Also, since graphs of genus g have separators of size $O(\sqrt{gn})$ [GH84], their chromatic polynomials can be computed in time $n^{O(\sqrt{gn})}$.

We also present results for a class of degree bounded[2] "near-planar" graphs. Informally, a **near-planar** graph with n nodes is a graph laid out on the plane with $O(n)$ edge crossovers. We show that the chromatic polynomial of such a graph can be computed in time $n^{O(\sqrt{n}\log n)}$.

The remainder of this paper is organized as follows. In Section 2, we consider the homomorphism problem for a planar source graph and an arbitrary target graph and show how this result can be used to compute the chromatic polynomial of a planar graph. Extensions to other classes of graphs with small separators are also indicated in that section. Section 3 presents results for near-planar graphs and Section 4 offers concluding remarks.

2. HOMOMORPHISM FROM A PLANAR SOURCE

In this section, we show how dynamic programming in conjunction with the planar separator theorem can be used to solve the homomorphism problem for a planar source. For ease of exposition, we first consider fixed target graphs. The target need not be planar.

2.1. Fixed Target

Let $G(V,E)$ be the given planar source with $|V| = n$ and $H(V',E')$ denote the fixed target with $|V'| = k$. Since H is not a part of the problem instance, k is a constant. Our goal is to determine whether there is a homomorphism from G to H and if so, to find one such homomorphism. We also determine how many distinct homomorphisms exist. Unless mentioned otherwise, throughout Section 2, $G(V,E)$ is a planar graph and A, B and C form a partition of V as given by the separator theorem for planar graphs. For any subset X of V, G_X will denote the subgraph of G induced on X. Given a homomorphism h from G to H, h_X denotes the restriction of h to X. Clearly, h_X is a homomorphism from G_X to H.

---

[1] Of course, a closed form expression for the chromatic polynomial of a tree is known [L68].

[2] By "degree bounded" we mean that the degree of each node is bounded by a constant d.

Let $V = \{v_1, v_2, ..., v_n\}$ denote the node set of G and $V' = \{v_1', v_2', ..., v_k'\}$ denote the node set of H. In order to formulate the homomorphism problem as a dynamic programming problem, we associate an integer variable x_i with node v_i in V $(1 \leq i \leq n)$. Each of these variables is constrained to take on a value from the set $\{1,2, ..., k\}$ so that for every homomorphism from G to H, there is a corresponding assignment of values to the variables. The interpretation of this constraint is the following: if x_i is assigned the value j, then the node v_i is mapped to the node v_j' by the corresponding homomorphism. The objective function $f(x_1,x_2,\ldots,x_n)$ to be maximized is given by,

$$f(x_1,x_2,\ldots,x_n) = \sum_{e=\{v_i,v_j\}\in E} f_e(x_i,x_j)$$

where $x_i \in \{1,2,\ldots,k\}$ $(1 \leq i \leq n)$, and $f_e(x_i,x_j) = 0$ if $\{x_i,x_j\} \in E'$ and $= -\infty$ otherwise. The maximum value of the objective function f is zero if and only if there is a homomorphism from G to H. Also note that there is a one-to-one correspondence between the set of assignments of values to $x_1, x_2, ..., x_n$ for which f is zero and the set of homomorphisms from G to H. Often we don't distinguish between a vertex v_i of V and the corresponding x_i.

Given a dynamic programming formulation, the associated **interaction graph** is defined as follows [Ro82]. For every variable we have a node and whenever two variables occur in the same term (the function f_e in the above formulation) of the objective function, there is an edge between the corresponding nodes. Clearly, the interaction graph corresponding to the above formulation is the source graph G itself. As was first observed in [LT80], whenever the interaction graph has small separators, the corresponding dynamic programming problem can be solved (asymptotically) much faster than exhaustive search.

Following some principles from [LT80], we give an algorithm for solving the homomorphism problems. We will need the following:

Definition 2.1: Let C be a subset of nodes of a source graph G and α an assignment of values to the nodes in C. Assignment α is called **legal** if and only if it corresponds to a homomorphism from G_C to H. A homomorphism h from G to H is called α-**constrained** if and only if $h(c) = \alpha(c)$ for every c in C.

Given a planar graph $G(V,E)$ with $|V| = n$, we can output the number of homomorphisms from G to fixed target graph H by the following algorithm COUNT:

> If n is small, solve by exhaustive search. Otherwise obtain sets A, B, and C from the planar separator theorem and consider each possible legal assignment α to nodes in C. For each α, recursively find the numbers N_{AC} and N_{BC} of α-constrained homomorphisms from $G_{A \cup C}$ and $G_{B \cup C}$ to H. Compute the products $N_{AC} \times N_{BC}$ for all α and output the sum.

COUNT is easily modified to output a homomorphism (if any). The correctness of Algorithm COUNT uses the following lemma.

Lemma 2.2: Let α be a legal assignment of values to nodes in C. If h_{AC} and h_{BC} are α-constrained homomorphisms from $G_{A \cup C}$ to H and from $G_{B \cup C}$ to H respectively then $h = h_{AC} \cup h_{BC}$ is an α-constrained homomorphism from G to H.

Proof: Omitted.

Lemma 2.2 shows that, for each legal assignment α of values to nodes in C, the union of *any* α-constrained homomorphism from $G_{A \cup C}$ to H with *any* α-constrained homomorphism from $G_{B \cup C}$ to H will result in an α-constrained homomorphism from G to H. It follows that, for each legal assignment α, the number of α-constrained homomorphisms from G to H is the *product* of the number of α-constrained homomorphisms from $G_{A\cup C}$ to H and the number from $G_{B\cup C}$ to H. Since Algorithm COUNT sums up the number of homomorphisms over all possible legal assignments, it counts all homomorphisms and we have:

Theorem 2.3: Algorithm COUNT correctly computes the number of homomorphisms from a planar source G to a fixed target H. □

We now consider the time complexity of the algorithm.

Theorem 2.4: Algorithm COUNT runs in time $2^{O(\sqrt{n})}$ where n is the number of nodes in G.

Proof sketch: Since $|C| = O(\sqrt{n})$ and each node of C is assigned a value from the set $\{1,2,\ldots,k\}$, the total number of assignments considered by the algorithm is $k^{O(\sqrt{n})} = 2^{O(\sqrt{n})}$. Using this observation, the argument to prove the running time is identical to that used in [LT80] for finding a maximum independent set in a planar graph. □

2.2. Variable Target

We now consider the extension of Algorithm COUNT to the case of variable targets. Let the target graph H have m nodes. The dynamic programming formulation of the homomorphism problem is identical to the fixed target case except that each variable can now take on any of m values as determined by reading H from the input. The number of assignments considered by the algorithm is thus $m^{O(\sqrt{n})}$. This observation in conjunction with the argument used in [LT80] leads to the following:

Theorem 2.5: The homomorphism problems for a planar source G with n nodes and a (variable) target H with m nodes can be solved in $m^{O(\sqrt{n})}$ time.

2.3. Computing the Chromatic Polynomial of a Planar Graph

Given a node labeled graph G, the **chromatic polynomial** of G, denoted by $P_G(\lambda)$, gives the number of ways of coloring the nodes of G using *at most* λ colors [L68]. In this section we show that the algorithm for counting the number of homomorphisms to a variable target can be used to compute the chromatic polynomial of a graph. Our algorithm runs in time $n^{O(\sqrt{n})}$ (equivalently, in time $2^{O(\sqrt{n}\log n)}$)

for planar graphs.

In defining the chromatic polynomial of a (labeled) graph, permutations of colors denote *distinct* node colorings [R84, Re68]. That is, each valid r-coloring is considered as a mapping from the node set of G to the set $\{1,2,\ldots,r\}$. As observed earlier, each such mapping corresponds to a homomorphism from G to K_r. Therefore, we have:

Observation 2.6: For any graph G, the value of $P_G(\lambda)$ evaluated at $\lambda = r$ is equal to the number of homomorphisms from G to K_r.

Recall that $P_G(\lambda)$ is a polynomial of degree n. It is well known that, given the values of a polynomial of degree n at $(n+1)$ distinct points, the coefficients of the polynomial can be determined through interpolation. Since the number of ways of coloring a graph with zero colors is zero, we know that the constant term in $P_G(\lambda)$ is zero. Therefore, it suffices to determine the values of $P_G(\lambda)$ for $\lambda = 1,2,\ldots,n$. We thus obtain the following:

Theorem 2.7: The chromatic polynomial of a planar graph can be determined in time $n^{O(\sqrt{n})}$.

The proof techniques easily extend to the following:

Theorem 2.8:

(1) The chromatic polynomial of a series-parallel graph can be computed in $O(n^4)$ time.

(2) The chromatic polynomial of a graph of fixed genus g can be computed in time $n^{O(\sqrt{gn})}$.

(3) The chromatic polynomial of a graph of bandwidth b (constant) can be computed in time $O(n^{b+2})$.

The proof of Part (1) of the above theorem uses the fact that series-parallel graphs satisfy a 2-separator theorem [HT86]; Part (2) uses the $O(\sqrt{gn})$-separator theorem for graphs of genus g [GH84] and Part (3) is based on the fact that graphs of bandwidth b satisfy a b-separator theorem.

3. NEAR-PLANAR GRAPHS

Near-planar graphs generalize the class of planar graphs. In order to define this class of graphs precisely, we begin with the definition of a *layout* of a graph.

Definition 3.1: Let $G(V,E)$ be a graph. A **layout** for G is a *planar* graph $G' = (V', E')$ together with a set S of **crossover nodes** and a function $f{:}E' \rightarrow E$ such that

(a) $S \cap V = \varnothing$,
(b) $V' = V \cup S$,
(c) Each node of S has degree 4, and
(d) For all $\{a,b\} \in E$, the set $\{e \in E' : f(e) = \{a,b\}\}$ is a simple path from a to b in G' which does not involve any other nodes in V.

Intuitively, G' is a drawing of G in the plane with the set S marking the places where two edges of G cross each other. Note that every planar graph is its own layout.

Definition 3.2: A graph G with n nodes is **α-near-planar** (for real number α) if and only if there is a layout G' for G such that the number of crossover nodes in G' is at most αn. We call a graph **near-planar** instead of α-near-planar when α is understood to have some fixed value.

Although the complexity of recognizing near-planar graphs is open, it is important to study this class of graphs as they are more likely to occur in practice than planar graphs. For example, many VLSI circuits are α-near-planar for some small α. We assume that the degree of each node of a near-planar graph is bounded by a constant. This assumption is necessary to achieve subexponential running times.

Since a near-planar graph may have $\Omega(n)$ crossovers, its genus can be $\Omega(n)$ [WB78]. Therefore, the application of the separator theorem of [GH84] (mentioned earlier) may lead to separators of size $\Omega(n)$. Therefore, we need a different technique to solve coloring and homomorphism problems for near-planar graphs. This technique from [SH88], is based on the notion of a *separator tree*.

Definition 3.3: Let $G(V,E)$ be a graph. Let T be a rooted tree with node set V' and let $\alpha{:}V \rightarrow V'$ map nodes of G onto the nodes of T. We write aPb for nodes $a,b \in V'$ if and only if a and b are on some common path to the root of T. For each $x \in V'$, define $Q(x) = \{a \in V : \alpha(a) = x\}$ to be the set of nodes *associated with* x. We call

(T,α) a **separator tree** for G if and only if

(i) $\alpha(a) P \alpha(b)$ for every $\{a,b\} \in E$ and
(ii) $Q(x) \neq \varnothing$ for all $x \in V'$, except for the root of T.

For each path p to the root of T, the **weight** of p, denoted by $w(p)$, is defined as

$$w(p) = | \{a \in V : \alpha(a) \text{ is on } p\} |$$

The **weighted depth** of (T,α) is the *maximum* weight of all paths in T. A proof of the following theorem appears in [SH88].

Theorem 3.4: Given a near-planar graph $G(V,E)$, with $|V| = n$, and a layout G' for G, a separator tree for G with weighted depth $O(\sqrt{n}\log n)$ can be obtained in polynomial time.

The importance of the separator tree is that using dynamic programming, several (NP-hard) problems for near-planar graphs can be solved in time exponential in the *weighted depth* of the separator tree rather than the number of nodes of G. The main idea of such algorithms is to start from the root of the separator tree and make assignments to the nodes associated with the root. This procedure is applied recursively to each child of the root. Since the assignments can be made to the children of the root independent of each other, the complexity is bounded by the cost of trying all assignments along a branch (an exponential in weighted depth) times the number of branches (at most n). This idea allows us to prove:

Theorem 3.5: Given a near-planar graph $G(V,E)$ and its layout G',

(1) Homomorphism problems for a fixed target can be solved in time $2^{O(\sqrt{n}\log n)}$.
(2) Homomorphism problems for a variable target with m nodes can be solved in time $m^{O(\sqrt{n}\log n)}$.
(3) The chromatic polynomial of G can be computed in time $n^{O(\sqrt{n}\log n)}$.

4. CONCLUSIONS

We have shown that a combination of dynamic programming and separator theorems is a powerful tool for solving a variety of node coloring and related problems for several classes of graphs. We have been able to apply these techniques in devising algorithms for coloring hypergraphs and also for edge coloring problems. These results will be presented in another paper.

5. REFERENCES

[Al87] Albertson,M.O., "Generalized Colorings", Proc. Japan-US Joint Seminar on Discrete Algorithms and Complexity, June 1986, pp 35-49. (These proceedings appear as Perspectives in Computing, Vol. 15, Academic Press, Inc., Edited by D.S.Johnson, et.al., Orlando, FL, 1987).

[FH86] Farber,M., Hahn,G., Hell,P. and Miller,D., "Concerning the Achromatic Number of Graphs", JCT Series B, Vol. 40, 1986.

[GH84] Gilbert,J.R., Hutchinson,J.P. and Tarjan,R.E., "A Separator Theorem for Graphs of Bounded Genus", J.Algorithms, Vol. 5, No. 3, Sept 1984, pp 391-407.

[GJ79] Garey,M.R. and Johnson,D.S., "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H.Freeman and Co., 1979.

[HT86] Hassin,R. and Tamir,A., "Efficient Algorithms for Optimization and Selection on Series-Parallel Graphs", SIAM J. Alg. Disc. Meth., Vol. 7, No. 3, July 1986, pp 379-389.

[L68] Liu,C.L., "Introduction to Combinatorial Mathematics", McGraw-Hill Publishing Co., 1968.

[LT79] Lipton,R.J. and Tarjan,R.E., "A Separator Theorem for Planar Graphs", SIAM J. of Appl. Math., Vol 36, No 2, April 1979, pp 177-189.

[LT80] Lipton,R.J. and Tarjan,R.E., "Applications of a Planar Separator Theorem", SIAM J. on Computing, Vol 9, No 3, August 1980, pp 615-627.

[MS81] Maurer,H.A., Sudborough,I.H. and Welzl,E., "On the Complexity of the General Coloring Problem", Information and Control, Vol. 51, 1981, pp 128-145.

[Mo87] Moser,L.E., "Graph Homomorphisms and the Design of Secure Computer Systems", Proc. 1987 IEEE Computer Society Conference on Security and Privacy, Oakland, CA, April 1987, pp 88-96.

[R84] Roberts,F.S., "Applied Combinatorics", Prentice-Hall, Inc., 1984.

[Re68] Read,R.C., "An Introduction to Chromatic Polynomials", JCT, Vol.4, 1968, pp 52-71.

[RH87] Ravi,S.S. and Hunt III,H.B., "An Application of Planar Separator Theorem to Counting Problems", Info. Proc. Lett., Vol. 25, No. 5, July 10, 1987, pp 317-321.

[Ro82] Rosenthal,A., "Dynamic Programming is Optimal for Nonserial Optimization Problems", SIAM J. Comput., Vol. 11, No. 1, Feb 1982, pp 47-59.

[SH88] Stearns,R.E. and Hunt III,H.B., "Power Indicies and Easier NP-complete Problems", in preparation.

[WB78] White,A.T. and Beineke,L.W., "Topological Graph Theory", Chapter 2 of "Selected Topics in Graph Theory", by L.W.Beineke and R.J.Wilson (Editors), Academic Press, 1978, pp 15-50.

Algorithms for Bisecting Planar Graphs

Thang Nguyen Bui
Andrew Peck

Computer Science Department
The Pennsylvania State University
University Park, Pa. 16802

Abstract

Let a planar graph G on n vertices be given together with a planar embedding $\widehat{G}$ of G. Let b be the bisection width of G and m be such that the embedding of each biconnected component in $\widehat{G}$ is at most m-outerplanar (we call such an embedding *m-outerplane-separable*). We give an algorithm to find an optimal bisection of G in time $O(\min\{b^2 n^3 2^{4.5b}, m^2 n^3 2^{3m}\})$.

An immediate corollary is that an optimal bisection is found in polynomial time for any planar graph with $O(\log n)$ bisection width or for which an $O(\log n)$-outerplane-separable embedding is available. This is the first such result for a subclass of planar graphs more general than trees.

The above results are not restricted to graph bisection. Given any $s \in [0, n]$, the same algorithm will find a minimal edge set which partitions the vertex set of the planar graph into two sets of size s and $n - s$.

1. Introduction

A *bisection* of a graph is a set of edges whose removal from the graph divides the n vertices of the graph into two disjoint sets of size $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$. An *optimal bisection* for a particular graph is a bisection containing a minimal number of edges. The *bisection width* of a graph is the number of edges in an optimal bisection. The *graph bisection problem* is the problem of finding an optimal bisection for a graph.

The problem of finding the minimum bisection of a graph arises in many graph algorithms which use the divide-and-conquer approach. A notable example is the VLSI placement and routing problem [BL84]. It is also observed that many other divide-and-conquer based algorithms run much faster on graphs in which a small bisection can be found, for example in the area of sparse matrix computation [GZ87], see also [LT80]. Despite its wide application, there is no known efficient algorithm for the graph bisection problem, since for general graphs the bisection problem is NP-complete. In fact, there are no known polynomial time approximation algorithms. Consequently, work has centered on finding heuristics with good average case behaviour. A number of methods are employed including a hill climbing heuristic [KL70], simulated annealing [KGV82], network flow [Bu87], and an approach based on the eigenvalues of the adjacency matrix [Bo87]. Only the last two approaches give provably good average case behaviour. Most of the other known heuristics have no theoretical analysis.

For special classes of graphs the situation is a little bit better. For example, an $O(n^2)$ time algorithm exists for trees [GM86][Mac78]. However, little is known for the class of planar graphs, including whether the problem is NP-complete. A motivation for studying this problem on planar graphs is that many examples found in practice involve planar graphs, e.g., VLSI placement. Another motivation is the close relation between the graph bisection problem and the vertex separator problem in planar graphs for which numerous applications have been shown [LT80]. In [R87] Rao considers similar problems, namely the edge separator problem and some of its variations on planar graphs. The results there, however, do not extend to the case of bisection since the sizes of the two subgraphs are allowed to be between 1/3 and 2/3 of the original graph.

In this paper we will consider the more general problem of partitioning a graph into two disjoint subgraphs of prescribed size. The graph bisection problem is a special case of this general problem. Specifically, let G be a graph on n vertices and $s \in [0, n]$ be given. An *s-partition* of G is a partition of the vertex set of G into two disjoint sets of size s and $n - s$. The *cutset* of an s-partition is the set of edges with endpoints in both parts of the partition, and the cardinality of the cutset is called the *cutsize* of the s-partition. An *optimal s-partition* is an s-partition whose cutsize is minimal for all s-partitions. In this notation a bisection of G is an $\lfloor n/2 \rfloor$-partition or an $\lceil n/2 \rceil$-partition of G.

To describe the main result in this paper we need the concept of a *k-outerplanar* embedding of a planar graph. Intuitively, a k-outerplanar embedding is an embedding consisting of disjoint outerplane graphs nested at most k deep. (An *outerplane* graph is a planar-embedded graph such that all vertices lie on the exterior face.) The main result may now be stated as follows.

Let an n-vertex planar graph G and $s \in [0, n]$ be given together with a planar embedding of G. Let m be such that each biconnected component of this embedding is at most m-outerplanar (we call such an embedding *m-outerplane-separable*) and b be the cutsize of an optimal s-partition of G. Then the optimal s-partition of G may be determined in time $O(\min\{b^2n^3 2^{4.5b}, m^2n^3 2^{3m}\})$.

An immediate corollary of this result is that if the cutsize of an optimal s-partition is $O(\log n)$ or if an $O(\log n)$-outerplane-separable embedding of G is available then an optimal s-partition of G can be found in polynomial time. In particular, if we let $s = \lfloor n/2 \rfloor$ or $s = \lceil n/2 \rceil$, then we have the results for graph bisection claimed in the abstract.

The remaining sections of the paper are as follows. In Section 2 we introduce the necessary terminology. In Section 3 we develop a table structure for representing optimal partitions of a graph. Sections 4 and 5 give algorithms for partitioning k-outerplane, k-outerplane-separable and general planar graphs. The conclusion and references are given in Sections 6 and 7.

2. Definitions and Terminology

In the remainder of this paper we will assume G to be a connected n-vertex plane graph. Extension of our results to disconnected graphs is easy. Also, since a linear time algorithm exists (see [HT74]) to find a planar embedding of a planar graph, the assumption that the input graph is plane does not affect our conclusions.

2.1. Vertex Level, k-Outerplanarity

Let G be an n-vertex plane graph. All vertices on the exterior face of G are *level* 1 vertices. Remove all vertices of level less than or equal to i, together with their incident edges. The vertices on the new exterior face are the *level* $(i+1)$ vertices.

A planar embedding of a graph is *k-outerplanar* if it has no vertices of level greater than k. A planar graph is *k-outerplanar* if it has a k-outerplanar embedding. Every planar graph is k-outerplanar for some k. The terms *outerplanar* and *1-outerplanar* are equivalent. In keeping with the familiar term *plane graph* a *k-outerplane graph* is a k-outerplanar embedding of a planar graph.

A *level edge* is an edge connecting two vertices at the same level in a k-outerplane graph. An *interlevel edge* connects a vertex at some level i to a level $(i-1)$ or level $(i+1)$ vertex. It is clear that a k-outerplane graph has only level and interlevel edges.

Consider a set of vertices in a k-outerplane graph which are connected by a path consisting only of level edges. A *level subgraph* is a maximal set of such vertices together with all level edges connecting them. A *level i subgraph* is a level subgraph containing level i vertices. If $i > 1$, then a level i subgraph is nested within and is connected via interlevel edges to exactly one level $(i-1)$ subgraph. Many level $(i+1)$ subgraphs may be nested within one level i subgraph. In general, a level subgraph consists of a collection of simple cycles (possibly with chords) connected by single edges and articulation points.

A *k-outerplane-separable* graph is a plane graph the biconnected components of which are k-outerplane. Intuitively, any non-trivial k-outerplane-separable graph looks like nested k-outerplane components each (except the outermost) sharing a single vertex with an enclosing level subgraph.

2.2. Vertex Partitions

Recall the definitions of an s-partition, cutset and cutsize in the introduction. For a particular partition we will arbitrarily label the two vertex sets *left side* and *right side*. Thus, it will make sense to speak of a vertex as being, for example, 'on the left side' of the partition. Once so labeled, a partition will be considered distinct from the partition obtained by exchanging the labels 'left' and 'right'. The number of vertices on the left side of a partition is called the *partition size*.

A *constrained partition* is a partition formed subject to the requirement that particular vertices must appear on the left and certain others on the right. The set of constraining vertices is called the *constraint set*. Given a constraint set, we will speak of an *assignment* of its vertices to the left and right sides of a partition, this assignment being the actual constraint imposed on the partition. Size and optimality for a constrained partition have the same meaning as in the unconstrained case above with the exception that some partition sizes may be impossible.

3. Slice Table

We describe a table structure which records optimal constrained solutions for a portion of a k-outerplane graph called a 'slice'. Our motivation is the adaptation in Section 4 of a dynamic programming scheme from [Ba83] to the bisection problem. For the sake of intuition we consider simply nested graphs in this section. Extension of the ideas to non-simply nested graphs is straightforward. By a simply nested graph we mean a k-outerplane graph in which each level subgraph is a simple cycle and for each $i \in [1, k]$ there is exactly one level i subgraph.

3.1. Slice Boundary and Slice

A *level i slice boundary*, or simply *slice boundary* when i is understood, is a set of i vertices, one from each of levels 1 to i, such that a line joining the vertices in order of level number does not intersect any edge which does not have an endpoint on the boundary. A *slicing* of a graph is a selection of slice boundaries such that (i) every level i vertex is part of a level i slice boundary; (ii) every level i slice boundary, $i > 1$, is the extension of some level $(i-1)$ slice boundary; (iii) a line drawn through the vertices of one boundary in order of level number does not cut a similar line drawn for any other boundary (two boundaries may coincide at points).

It is natural to speak of a *clockwise* traversal of a level subgraph in a simply nested graph since each such subgraph is a simple cycle. The endpoints of an edge of a level subgraph are labeled *ccw* and *cw* in such a way that a clockwise traversal of the level subgraph which starts at the *ccw* endpoint of an edge traverses that edge immediately.

Given a slicing of a graph, select two level i slice boundaries and label the boundaries arbitrarily *ccw* and *cw*. A *level i slice* consists of (i) all vertices encountered in a *cw* traversal of any level subgraph beginning at the *ccw* boundary and ending at the *cw* boundary; (ii) all edges with both endpoints in the slice except for any level edge with its *ccw* endpoint in the *cw* slice boundary and its *cw* endpoint in the *ccw* slice boundary. Two slices are *adjacent* if the *cw* boundary of one is the *ccw* boundary of the other. In this case, we modify the definition of slice so that any interlevel edge with both endpoints in a common boundary is arbitrarily included in one slice or the other, but not both.

3.2. Partition Vector

A *partition vector* for a graph G on m vertices is an $(m+1)$-vector P, where $P[i]$, $i \in [0, m]$, is the cutsize of an optimal i-partition of G. Let A and B be two sets of k vertices of G. Designate the vertices in $A \cup B$ as a constraint set for partitions of G. Let X be a k-bit vector such that $X[i] = 1$ $(X[i] = 0)$ if the i^{th} element of A is constrained to be on the left (right) side of the partition. Let a k-bit vector Y be defined similarly for B. A *constrained partition vector* P_{XY} is defined as in the unconstrained case except that $P_{XY}[i]$ is now the optimal i-partition obtainable subject to the constraints represented by X and Y. If a constrained i-partition is impossible, then let $P_{XY}[i] = \infty$. Since A and B are not required to be disjoint, the values of X and Y may represent inconsistent assignments of some vertex in $A \cap B$. In this case let $P_{XY}[i] = \infty$ for all i.

In the above definition of a constrained partition vector replace G by a slice S of G. Let the constraint sets A and B be the level 1 to level j, $j \leq i$, vertices of the *ccw* and *cw* slice boundaries of S respectively. Then X and Y are both j-bit vectors and P_{XY} is a constrained partition vector for S. A *slice table* for S is an arrangement of the 2^{2j} possible constrained partition vectors for S in a square table indexed by X and Y. That is, if X_B, Y_B are the binary values represented by the bit vectors X and Y, then the $(X_B, Y_B)^{\text{th}}$ table entry is the constrained partition vector P_{XY}. Note that half the information in the table is redundant since $P_{XY}[s] = P_{\bar{X}\bar{Y}}[n-s]$ where $\bar{X}, \bar{Y}$ is the bit complement of X, Y. In the special case of a constraint set consisting of a single vertex v it will be convenient to refer to a constrained partition vector P_v which, because of redundancy, contains all the information of a $2^1 \times 2^1$ table.

3.3. Table Operations

We state three lemmas describing operations on slice tables - *merger, extension,* and *constraint removal.* We prove the lemma for the table merger operation in detail. Proofs of the remaining two are similar and are omitted.

Lemma 3.1. *(Table merger) Let S_1 and S_2 be adjacent level i slices for which slice tables have been computed. A slice table for the slice $S = S_1 \cup S_2$ may be computed from the tables for S_1 and S_2 in time $O(2^{3i}m^2)$ where m is the number of vertices in S.*

Proof: Suppose w.l.o.g. that the *cw* boundary of S_1 corresponds to the *ccw* boundary of S_2. Let X, Y, and Z be bit vectors reflecting left/right assignments of the vertices of, respectively, the *ccw* boundary of S_1, the *cw* boundary of S_2, and the set of vertices which are both the *cw* boundary of S_1 and the *ccw* boundary of S_2. In other words, X and Y are the constraints for the *ccw* and the *cw* boundaries of S. Denote by P^1_{XZ}, P^2_{ZY}, and P_{XY} constrained partition vectors for S_1, S_2, and S, respectively. Denote by z the number of 1's in the bit vector Z. Thus, given a partition of the vertices of S, z is the number of vertices of the common boundary of the two sub-slices S_1 and S_2 which lie on the left side of the partition.

To see that the tables for S_1 and S_2 contain sufficient information to construct a table for S observe that an optimal constrained partition of S induces optimal constrained partitions of S_1 and S_2. Further, if a partition of S_1 and a partition of S_2 have the same left/right assignments of the common boundary vertices, then they combine to form a partition of S. The following two claims, which are straightforward to prove, formalize the argument. Let the number of vertices in S_1 and S_2 be m_1 and m_2, respectively and let X, Y, Z, and z be as above.

Claim 1. *Let* $x_1 \in [0, m_1]$, $x_2 \in [0, m_2]$, *and* $x \in [0, m]$ *be such that* $x_1 + x_2 - z = x$, *then* $P^1_{XZ}[x_1] + P^2_{ZY}[x_2] \geq P_{XY}[x]$.

Claim 2. *Given* x, *there exist* $x_1 \in [0, m_1]$, $x_2 \in [0, m_2]$ *and a bit vector* Z *such that* $x_1 + x_2 - z = x$ *and* $P^1_{XZ}[x_1] + P^2_{ZY}[x_2] = P_{XY}[x]$.

Claims 1 and 2 imply that in order to determine $P_{XY}[x]$ from the tables for S_1 and S_2 we need only find values for Z, x_1, x_2 which minimize $P^1_{XZ}[x_1] + P^2_{ZY}[x_2]$ and such that $x = x_1 + x_2 - z$. Thus, to compute a table representing the merger of the slices S_1 and S_2 it is only necessary to scan each entry P^1_{XZ} once for each entry P^2_{ZY}. If m_1, m_2 are the numbers of vertices in S_1 and S_2, then this can be done in time proportional to $2^{3i} m_1 * m_2$ which is $O(2^{3i} m^2)$. ∎

The remaining two operations can be proved by similar considerations.

Lemma 3.2. *(Table extension) Let* S *be a level* i *slice containing* m *vertices for which a slice table has been computed. Let* v *be a level* $(i+1)$ *vertex such that the boundaries of* S *may be extended to include* v *and such that there is no edge from* v *to a level* i *vertex in* S *unless it is in one of the boundaries. A table for the extended slice may be computed from the table for* S *in time* $O(2^{2i} m)$.

Lemma 3.3. *(Constraint removal) Let* S *be a level* i *slice containing* m *vertices for which a* $2^i \times 2^i$ *slice table has been computed. A* $2^j \times 2^j$ *slice table,* $j < i$, *representing the same slice constrained only by the first* j *vertices of each border may be computed in time* $O(2^{2i} m)$.

4. Partitioning k-Outerplane Graphs

In this section we describe how to partition k-outerplane and k-outerplane-separable graphs in polynomial time. We also show that the method used extends to graphs with weighted edges.

In [Ba83] Baker provides a general method for building tables of solutions for various combinatorial optimization problems on k-outerplane graphs. We provide a brief description of Baker's method here and refer the reader to the original paper for details.

Baker's method works generally with min-/maximization problems on the vertices of k-outerplane graphs in which the quantity to be optimized is such that a solution for the entire graph may be computed directly from constrained optimal solutions on subgraphs which are disjoint but for boundary vertices. A problem dependent table structure records optimal constrained solutions on slices of a graph. A problem independent scheme then builds up a solution table for the entire graph. The scheme works by first defining trivial slices for which tables are easily computed and then extending and merging these slices (while carrying out the corresponding operations on the associated tables) until a single 'slice' and table represents the entire graph. The time required is linear in the number of vertices multiplied by the time for the most costly table operation for a level k slice table.

The table structure and operations of the previous section have been defined so that Baker's method may be applied to solve the graph partition problem on k-outerplane graphs. The most costly of the operations is table merger requiring time $O(2^{3i}m^2)$, where m is the number of vertices in the combined slices and i is the maximum level of the slice. Since $m \leq n$ and $i \leq k$, the following theorem is an immediate result of the above discussion.

Theorem 4.1. *(Partition of k-outerplane graphs.) Let G be an n-vertex k-outerplane graph and $s \in [0, n]$ be given. An optimal s-partition of G may be found in time $O(2^{3k}n^3)$.*

The algorithm returns a single partition vector P for the entire graph, where $P[s]$ is the cutsize of an optimal s-partition of the graph. If the actual partition is required, then it may be found by preserving the tables and retracing the steps of the computation. Space required by the tables is that for $2^{2k}n$ partition vectors which is $O(2^{2k}n^2)$.

The above scheme may be made to return a constrained partition vector P_v representing partitions constrained by a single vertex v on the exterior face (at the start select the appropriate vertex and at the conclusion remove all constraints except the constraint on this vertex). This is important to the following theorem which follows from the observation that biconnected components may be processed separately to yield tables constrained only by articulation points.

Theorem 4.2. *(Partition of k-outerplane-separable graphs.) Theorem 4.1 holds for G an n-vertex k-outerplane-separable graph.*

For graphs with weighted edges *cutsize* and *optimal cutsize* may be redefined in terms of the total weight of edges in the cutset. To modify the algorithm to accomodate such graphs we need only change procedures in which the effect of cutting an individual edge is considered as is the case in table extension. The cost of cutting an edge is now simply its weight rather than 1. Running time is unaffected. The space requirement is unaffected so long as we accept a word model in which individual elements in a partition vector may have arbitrary size. The following corollary is the result of these observations.

Corollary 4.3. *Theorems 4.1 and 4.2 hold for graphs with weighted edges.*

5. Partitioning Planar Graphs With Small Optimal Cutsize

In Section 4 we showed how to find an optimal s-partition of an n-vertex graph G having m-outerplane biconnected components. We obtained a polynomial time algorithm when $m = O(\log n)$. In this section we show that we can still get a polynomial time algorithm when m is not $O(\log n)$ provided that the cutsize of the optimal s-partition is small. Note that a small cutsize does not imply that m must be small, or vice versa.

The main idea here is to construct a series of edge-weighted k-outerplane-separable subgraphs G_i^* of a plane graph G which isolate every cutset no larger than k of G. By applying this construction together with Theorem 4.2 for increasing values of k we obtain an optimal partition of the graph.

5.1. Isolating Optimal Cutsets

To isolate a cutset with cutsize k or less we construct a set of weighted subgraphs G_i^*, $i \in [0, k]$ with the property that an edge set S, $|S| \leq k$, is the cutset of an optimal s-partition of G if and only if it is the cutset of an optimal s-partition of at least one of the subgraphs. The main idea of the construction is as follows. For any level $(j+2)$ subgraph, $j = i \bmod (3k+3)$, remove all but one of the edges connecting it to the enclosing level $(j+1)$ subgraph. In addition give weight ∞ to all edges with endpoints in a level $(j+1)$ or $(j+2)$ subgraph, and weight 1 to the rest. The result is a weighted $(3k+3)$-outerplane-separable graph G_i^*. We will show that if an optimal s-partition of G has cutsize no greater than k, then for any optimal s-partition of G there is some l such that the cutset is contained completely among the weight 1 edges of G_l^*. Furthermore, it is straightforward to show that the same set of edges is also the cutset for an optimal s-partition of G_l^*.

Lemma 5.1. *Let G be an m-outerplane graph and $s \in [0, n]$. If an optimal s-partition of G has cutsize $w \leq k$, then an edge-weighted subgraph G^* of G may be constructed such that G^* is $(3k+3)$-outerplane-separable and such that an optimal s-partition of G^* is an optimal s-partition of G.*

Proof: If $k \geq m/3 - 1$, then G is $(3k+3)$-outerplane and we are done. Assume that $k < m/3 - 1$.

Let S be the cutset for some optimal s-partition of G. There are no more than k edges in S. For each $i \in [0, k]$ define J_i, G_i, H_i as follows.

- $J_i = \{j \mid j = 3i \bmod (3k+3), j \in [0, m-2]\}$
- G_i is the subgraph consisting of G minus all level $(j+1)$ and $(j+2)$ subgraphs and their incident edges, for all $j \in J_i$.
- H_i is the set of all edges of G which are not in G_i.

For each $i \in [0, k]$ construct a weighted subgraph G_i^* as follows. Let $j \in J_i$. For every level $(j+2)$ subgraph of G remove all but one interlevel edge connecting it to the enclosing level $(j+1)$ subgraph. Give weight ∞ to all edges of G_i^* which are in the set H_i. Give weight 1 to all other edges. The result is an edge-weighted $(3k+3)$-outerplane-separable graph. Call the edges with weight ∞ *heavy* edges and the remaining edges *light* edges. Observe that the light edges are exactly the edges of G_i and that only heavy edges have endpoints in the level $(j+1)$ and $(j+2)$ subgraphs.

The $k+1$ edge sets H_i are clearly disjoint. Thus, there is some $l \in [0, k]$ such that H_l contains no edges of S. It follows that all of the edges of S are in G_l. By our previous observation all the edges of S are light edges in G_l^*.

The following claims will complete the proof of Lemma 5.1.

Claim 1. *A partition of the vertices of G which separates the endpoints of an edge a which is not in G_i^* will also separate the endpoints of some heavy edge in G_i^*.*

An edge which is not in G_i^* must connect a level $(j+1)$ and a level $(j+2)$ subgraph, $j \in J_i$. In G_i^* any two such level subgraphs contain only heavy edges and are joined by a single heavy edge. Thus there is a path in G_i^* joining the endpoints of a and consisting entirely of heavy edges. Any partition of the vertices of G and, hence, of G_i^* which separates the endpoints of a must also cut some edge on this path.

The next four claims are not difficult to show and we omit the proofs.

Claim 2. *Any cutset of G_i^* which includes only light edges is also a cutset for G.*

Claim 3. *Any cutset of weight less than ∞ of G_i^* is also a cutset of G.*

Claim 4. *An optimal s-partition of any G_i^* has cutsize no less than w.*

Claim 5. *S is an optimal cutset for G_l^*, for some $l \in [0, k]$.*

To obtain the G^* of Lemma 5.1 we need only construct the $k+1$ graphs G_i^* and take as G^* that G_i^* with an s-partition of minimal cutsize. ∎

In the case that G is biconnected, a more careful analysis in the proof of Lemma 5.1 shows that we need to construct only half as many subgraphs in order to isolate an optimal cutset. Moreover, the subgraphs are only $(1.5k+1.5)$-outerplane-separable. This gives the following lemma.

Lemma 5.2. *If G is biconnected, then Lemma 5.1 holds if G^* is $(1.5k+1.5)$-outerplane-separable.*

5.2. The Algorithms

We describe three algorithms which use the construction of Lemma 5.2. Algorithm 5.1 partitions an m-outerplane biconnected graph. Algorithm 5.2 does the same for an m-outerplane-separable graph. Algorithm 5.3 uses Algorithm 5.2 to partition a general plane graph and gives us the proof of the main theorem of the paper.

Let G be m-outerplane and biconnected. Let $k \leq |E|$, E the edges of G. The following algorithm finds an optimal s-partition of G for all s such that the cutsize of the optimal s-partition is no more than k.

Algorithm 5.1

Input: G biconnected and m-outerplane; $k \leq |E|$; v a vertex on the external face of G.

Output: A vector P_v.

Step 1. If $k \geq 2m/3 - 1$ then apply Theorem 4.1 returning a partition vector P_v for the entire graph, otherwise do Steps 2-4.

Step 2. Create $\lfloor k/2 \rfloor + 1$ edge-weighted $(1.5k+1.5)$-outerplane-separable graphs G_i^* as described in Lemma 5.2.

Step 3. Use Theorem 4.2 to obtain a partition vector P_v^i for each G_i^*.

Step 4. Return a vector P_v in which $P_v[s] = \min_{i \in [0, \lfloor k/2 \rfloor]} P_v^i[s]$. ∎

Observe that if Steps 2 to 4 are executed, then P_v is not strictly a partition vector for G since $P_v[s]$ is guaranteed to be optimal only if it is no more than k. If $P_v[s] > k$ then the optimal s-partition has cutsize greater than k, but has not necessarily been found.

Correctness of Algorithm 5.1 follows from consideration of Theorems 4.1 and 4.2, Corollary 4.3, and Lemma 5.2. If Steps 2-4 are executed, then Step 2 dominates the running time. Assuming an appropriate data structure for the input graph G, the construction of G_i^* proceeds in linear time. Thus, by Theorems 4.2 and 4.3, the time requirement is $O(\min\{2^{3m}n^3, k2^{4.5k}n^3\})$.

Algorithm 5.1 can be extended without increasing the running time to handle input graphs which are m-outerplane-separable. The extension is analogous to Theorem 4.2. In addition, the extended algorithm will remove the final constraint v. Let this extended algorithm be Algorithm 5.2. We are now ready to give an algorithm to partition general planar graphs.

Algorithm 5.3

Input: G, an n-vertex planar graph; s, the size of the partition sought.

Output: An optimal s-partition.

Step 1. Obtain a planar embedding of G, determine m such that this embedding is m-outerplane-separable.

Step 2. For increasing values of k apply Algorithm 5.2 with input G, k, returning vector P until $P[s] = k$ or $k = 2m/3 - 1$;

Step 3. If Step 2 does not return an optimal s-partition, then apply Theorem 4.2 to the full graph G to find an optimal s-partition. ∎

If an optimal s-partition of G has cutsize $b < 2m/3$ then Step 2 repeats b times. The running time for the i^{th} repetition of Step 2 is $O(i2^{4.5i}n^3)$. If Step 3 is not executed, then the final cost is $O(b^2 2^{4.5b}n^3)$, else the cost is $O(2^{3m}n^3)$ by application of Theorem 4.2, since it must be that $b \geq 2m/3 - 1$. Since we must account for the $2m/3-1$ unsuccessful iterations the final running time is $O(m^2 2^{3m} n^3)$. Algorithm 5.3 gives us the main theorem of the paper: .

Theorem 5.1. *Let an n-vertex planar graph G and $s \in [0,n]$ be given together with a planar embedding of G. Let m be such that this embedding is m-outerplane-separable and b be the cutsize of an optimal s-partition of G. Then the optimal s-partition of G may be determined in time* $O(\min\{b^2 n^3 2^{4.5b}, m^2 n^3 2^{3m}\})$.

Corollary 5.1 follows from Theorem 5.1 by substituting $c \log n$ for b:

Corollary 5.1. *Let G be an n-vertex planar graph and $s \in [0,n]$ be given. If the cutsize of an optimal s-partition is $O(\log n)$ or if an $O(\log n)$-outerplane-separable embedding of G is available, then an optimal s-partition of G may be found in polynomial time.*

Corollary 5.2 is a result of the observation that a bisection of a graph is an $\lfloor n/2 \rfloor$-partition or an $\lceil n/2 \rceil$-partition.

Corollary 5.2. *Let G be an n-vertex planar graph. If G has $O(\log n)$ bisection width or if an $O(\log n)$-outerplane-separable embedding of G is available, then an optimal bisection of G may be found in polynomial time.*

6. Conclusions

In this paper we showed that an optimal s-partition of a planar graph may be found in polynomial time if the cutsize of the optimal partition is $O(\log n)$ or if an embedding of the graph is available in which the embedding of each biconnected component is $O(\log n)$-outerplanar. Letting $s = \lfloor n/2 \rfloor$ or $s = \lceil n/2 \rceil$ we have the same result for the graph bisection problem.

Our method suggests some avenues for further study:

1. It would be useful to find a different method of constructing the subgraphs of Section 5.1 using a k smaller than b while preserving the essential property that some optimal cutset is guaranteed to be contained in some subgraph. In particular, if k can be held at most logarithmic in the size of the graph for some subclass of planar graphs, then a polynomial time algorithm for finding optimal s-partitions would result for that subclass.
2. The running time of our algorithms can be improved if we know how to efficiently generate a k-outerplanar embedding of a k-outerplanar graph. This is an open problem for general k-outerplanar graphs. In fact, by Theorem 4.2 it would be sufficient that we could do this for the case of biconnected planar graphs. An easy solution exists if the graph is 3-connected since it is known that such a graph has a unique embedding up to the choice of the exterior face [W31].
3. It seems that the the running time of our algorithms can be improved by considering the order of processing the slices. This is because for some graph embeddings the processing order will affect the total running time of the table merger operations.

7. References

[Ba83] Baker, B., "Approximation Algorithms for NP-Complete Problems on Planar Graphs," FOCS 1983, pp. 265–273.

[BL84] Bhatt, S. N. and F. T. Leighton, "A Framework for Solving VLSI Graph Layout Problems," Journal of Computer and System Sciences, Vol. 28, No. 2, 1984.

[Bo87] Boppana, R., "Eigenvalues and Graph Bisection: An Average-Case Analysis," Proceedings of the Symposium on Foundation of Computer Science, 1987, pp. 280–285.

[Bu87] Bui, T., S. Chaudhuri, T. Leighton, and M. Sipser, "Graph Bisection Algorithms With Good Average Case Behavior," Combinatorica, Vol. 7, No. 2, 1987, pp. 171–191.

[GZ87] Gilbert, J. R. and E. Zmijewski, "A Parallel Graph Partitioning Algorithm for A Message-Passing Multiprocessor," Cornell Computer Science Technical Report, TR 87-803, January 1987.

[GM86] Goldberg, M. and Z. Miller, "A Parallel Algorithm for Bisection Width in Trees," Unpublished Notes, 1986.

[HT74] Hopcroft, J. E. and R. E. Tarjan, "Efficient Planarity Testing," JACM, 21(4), 1974 pp. 549–568.

[KL70] Kernighan, B. W. and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," The Bell System Tech. J., Feb. 1970, pp. 291–307.

[KGV82] Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, No. 4598, May 13, 1983, pp. 671–680.

[LT79] Lipton, R. J. and R. E. Tarjan, "A Separator Theorem for Planar Graphs," SIAM J. Computing, Vol. 36, No. 2, April 1979. pp. 177–189.

[LT80] Lipton, R. J. and R. E. Tarjan, "Application of a Planar Separator Theorem," SIAM J. Computing, Vol. 9, No. 3, Aug. 1980, pp. 615–627.

[Mac78] MacGregor, R. M., "On Partitioning a Graph: A Theoretical and Empirical Study," Ph.D. Thesis, University of Berkeley, 1978.

[R87] Rao, S., "Finding Near Optimal Separators in Planar Graphs," FOCS 1987, pp. 225–237.

[W31] Whitney, H., "A Theorem on Graphs," Annals of Mathematics, 32, 1931, pp. 378–390.

A BETTER CLIQUE-PARTITIONING ALGORITHM*

J. BHASKER+ and TARIQ SAMAD
Honeywell CSDD, 1000 Boone Avenue North, Golden Valley, MN 55427

INTRODUCTION

We present two new efficient clique-partitioning algorithms. The *clique-partitioning* of a graph refers to the (NP-complete) problem of finding the smallest number of cliques (fully connected subgraphs) in a graph such that every vertex in the graph is represented in exactly one clique. Since the clique-partitioning of a graph is equivalent to the coloring of the complement of the graph, any coloring algorithm can also be used to clique-partition a graph (and vice versa).

A good comparison of the major coloring algorithms is given in [2]. Based on the conclusions reported therein, we compare our algorithms with the INTERSEQCOLORING algorithm, which is a sequential coloring algorithm with interchanges of colors allowed. There are two variants of INTERSEQCOLORING: the order in which the vertices are processed can be largest-first (COLOR-LF) or smallest-last (COLOR-SL). We also compare our algorithms with the clique-partitioning algorithm described in [3] (here referred to as TSENG). The TSENG algorithm constructs maximal cliques one at a time. The edge that is best connected is used to initiate a clique that is then expanded until the clique is maximal. Only then is a new clique initiated.

TWO NEW ALGORITHMS

Both our algorithms, referred to as METHOD-1 and METHOD-2, are derived from the basic algorithm shown in Figure 1. *METHOD-1:* The first algorithm is identical to the basic algorithm, with the clarification that if more than one vertex is identified in Step 3, an arbitrary choice is made. *METHOD-2:* In Step 3 of the basic algorithm described in Figure 1, if more than one vertex is identified as a candidate for *N2*, we choose one that has a common neighbor with the edge (*N1*, *N2*). If no candidate *N2* has a common neighbor with (*N1*, *N2*), an arbitrary choice is made.

Unlike TSENG, our algorithms construct cliques in parallel. Furthermore, the *least* connected vertices are considered first. The time complexity of METHOD-1 is $O(n^2)$ while that of METHOD-2 is $O(n^3)$.

PERFORMANCE COMPARISONS

We have implemented the five algorithms COLOR-SL, COLOR-LF, TSENG, METHOD-1 and METHOD-2 in Pascal. The run-time information for the coloring algorithms includes the time spent complementing the original graph before running the coloring algorithm. For comparison, we used five sets of ten randomly generated graphs. All graphs had 50 vertices. The number of edges in a graph was constant within a set and varied from 100 to 1000 across sets. The results indicated that METHOD-1 and METHOD-2 have the least percent deviation from optimal, with TSENG doing the worst. However, as the connectivity is increased, COLOR-SL and COLOR-LF become comparable, and eventually better than METHOD-1 and METHOD-2. This is clearly plausible since a large connectivity graph for clique-partitioning is a sparse graph for coloring and any good coloring algorithm will perform close to optimally.

To verify that these results held over a wider range of graphs, we tested all algorithms on graphs of 10 and 100 vertices, varying the number of edges for each case between 10 and 90 percent of full connectivity. It appears that as the size of the graph increases, our algorithms perform increasingly better relative to the coloring algorithms. Results also indicate that METHOD-2 performs better, albeit marginally, than METHOD-1; that TSENG produces the worst results; and that among the coloring algorithms, COLOR-LF performs better than COLOR-SL. Figure 2 shows the results of functional comparison for graphs of 50 and 100 vertices. For example, with ten random graphs with 100 vertices and 30% connectivity, 27.4 number of cliques were obtained on an average by METHOD-1.

---

* This paper results from work done on contract no. F33615-85-C-1861 from Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Aeronautical Systems Division (AFSC), United States Air Force, Wright-Patterson AFB, Ohio 45433-6543.

+ Current address: AT&T Bell Laboratories, 1247 S. Cedar Crest Blvd., Allentown, PA 18103

Figure 3 shows the run times of the various algorithms obtained when executed on an Apollo DN330 workstation. We see that METHOD-1 and METHOD-2 are fast algorithms. Both the coloring algorithms are slow because of the fact that they have to complement the graph before coloring can proceed. It is difficult to compare TSENG's run times since we used different data structures than what was originally proposed. The plot in Figure 4 illustatres the time complexity superiority of our algorithms. The curves for METHOD-1 and METHOD-2 are overlapped since they are very close to each other. The sparser the graph an algorithm is applied on, the faster the execution. Since the coloring algorithms are operating on the complement of the graph, their run times decrease with increasing connectivities.

A more detailed description of this research is available in [1].

REFERENCES

1. Bhasker J. and T.Samad, *A Better Clique-Partitioning Algorithm*, Technical Report, Honeywell CSDD, 1000 Boone Ave. N., Golden Valley, MN 55427, 1988.
2. Syslo M.M., N. Deo and J.S. Kowalik, *Discrete Optimization Algorithms*, Prentice Hall Inc., Englewood Cliffs, NJ, 1984.
3. Tseng C.J., *Automated Synthesis of Data Paths in Digital Systems*, Ph.D. dissertation, Dept. of ECE, CMU, Pittsburgh, April 1984.

1. If all vertices have zero degree, go to Step *7*.
2. Choose a non-zero degree vertex with the smallest degree. Call this *N1*.
3. Of all the vertices connected to *N1*, choose the one with the smallest degree. Call this *N2*.
4. Cluster *N1* and *N2* into a new compound vertex *N3*.
5. Modify the graph by replacing vertices *N1* and *N2* with the new vertex *N3*. A vertex *Nx* is connected to *N3* in the new graph iff *Nx* was connected to both *N1* and *N2* in the old graph. The degrees of affected vertices are updated.
6. Go to Step *1*.
7. Any vertices not already clustered are made into singular clusters.

Figure 1. The basic algorithm.

100 vertices	% connectivity				
	10	30	50	70	90
METHOD-1	42.2	27.4	18.8	12.9	7.1
METHOD-2	41.8	26.9	18.8	13.1	6.9
COLOR-SL	44.4	28.1	19.0	13.4	6.9
COLOR-LF	43.1	28.0	18.7	12.8	7.1
TSENG	44.7	29.3	20.3	15.1	8.6

50 vertices	% connectivity				
	10	30	50	70	90
METHOD-1	24.4	15.6	11.4	8.2	5.0
METHOD-2	24.4	16.2	11.4	8.5	5.0
COLOR-SL	25.4	16.6	12.0	8.3	4.7
COLOR-LF	24.8	16.6	11.2	8.0	4.7
TSENG	27.2	16.6	12.6	9.3	5.5

Figure 2. Performance of algorithms with varying connectivities. Entries denote average number of cliques obtained over 10 random graphs.

50 vertices	% connectivity			
	10	30	50	60
METHOD-1	.071	.091	.100	.104
METHOD-2	.072	.092	.102	.105
COLOR-SL	1.68	.700	.291	.186
COLOR-LF	1.76	.665	.302	.189
TSENG	.421	.967	1.093	.970

Figure 3. CPU run times (in seconds).

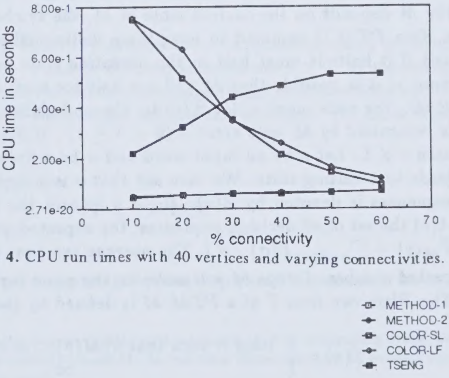


Figure 4. CPU run times with 40 vertices and varying connectivities.

Adaptive and Optimal Randomization of Algorithms in P: A Case Study

Wolfgang H. Janko, Andreas Geyer-Schulz, Alfred Taudes

Department of Applied Computer Science
Institute of Information Processing and Information Economics
University of Economics and Business Administration, Vienna, Austria

September 21, 1988

Abstract

The introduction of learning in randomized algorithms solving without error is investigated. The results of using selected methods for the adaptive randomization of a search algorithm in P are presented and discussed.

1 Introduction

The concept of *probabilistic algorithms* was introduced as a distinguished concept in the design of algorithms by [Rabin]. In his paper he introduces a primality testing algorithm and a nearest neighbour search algorithm. These two algorithms are conceptually very different. Wheras the primality testing algorithm uses sampling to eventually confirm the compositeness of a given number and is in its structure similar to the algorithm of [Solovay and Strassen], the nearest neighbour search algorithm always finds the solution of the problem with no error. Unlike the case of the primality testing algorithm there is no such arbitary small but positiv *probability of error*.

Algorithms which use some kind of randomization but yield the problem's solution with no probability of error or any other uncertainty shall be called *randomized* algorithms. Other types of probabilistic algorithms are *Las Vegas* and *Monte Carlo* algorithms. Both of them induce some uncertainty in the result of the problem's solution. To make these concepts clear, we use the definition of complexity classes in the class of problems solved by probabilistic Turing machines. We give an informal introduction to a probabilistic Turing machine following mainly [Yao 1977a]. For a more detailed discussion readers are referred to [Gill] and the references therein. A *probabilistic one-tape Turing machine (PTM)* M consists of a finite control, a read-write head on a 1-dimensional tape, and a *random symbol generator* (*RSG*) capable of generating integers i between 1 and i_0 with fixed probabilities p_i ([Gill] uses $i_0 = 2$ and $p_i = 0.5$). Before each move of the *PTM* the *RSG* produces a random symbol i, and the action of the M depends on the current state of M, the symbol of the tape being read, and the random symbol i. Our *PTM* is assumed to have three distinguished states: q_0, q_1 and q_2. The machine starts in q_0 and if it halts it must halt in the *accepting* state q_1 or in the *rejecting* state q_2. For a given input word v, it is possible that M will not halt for some infinite sequence of random symbols generated by *RSG*. For each input v, let $\beta_i(v)$ be the probability that M halts in state q_i $(i = 1, 2)$. A language L is *recognized by M with error* λ $(0 < \lambda \leq 1)$, if $\beta_1(v) \geq 1 - \lambda$ for each $v \in$ L, and $\beta_2(v) \geq 1 - \lambda$ for each $v \notin L$. Let v be an input word and σ be a finite sequence of integers generated by the *RSG* that leads to a halting state. We then say that σ is a decision sequence for v on M . The set of all decision sequences is denoted by $A(v)$. If $\mid \sigma \mid$ denotes the length of the sequence, $P(\sigma)$ the probability and $A(v)$ the set of all decision sequences, the expected number of steps M will make for input v is equal to $\overline{T}_M(v) = \sum_{\sigma \in A(v)} P(\sigma) \cdot \mid \sigma \mid$. The *average run time* of M for inputs of length n is defined to be the expected number of steps M will make for the worst input of length n: $\overline{T}(M, n) = max\{\overline{T}_M(v) \mid\mid v \mid = n\}$. The *Blum run time* T of a *PTM* M is defined by (see [Gill])

$$T(v) = \begin{cases} \text{least } n \text{ such that } p\{M(v) = \phi(v) \text{ in times}\} > 0.5 & \text{if } \phi(v) \text{ is defined,} \\ \infty & \text{if } \phi(v) \text{ is undefined} \end{cases}$$

where $\phi(v)$ is defined by

$$\phi(v) = \begin{cases} y & \text{if } p\{M(v) = y\} > 0.5, \\ \text{undefined} & \text{if no such } y \text{ exists} \end{cases}$$

and M computes the characteristic function of the language L. The run time used for the analysis of algorithms is very often the average run time which is especially reasonable for PTM with *bounded* error probability.

In the terminology used furtheron we shall emphasize the close connection of the field described with statistical theory by using a terminology close to that of statistics. Let M denote the PTM with $v \in \{0,1\}^*$. Let L denote a set of input strings v on which M halts. If $M(v)$ is the result of the reaction of M on v let $M(v)$ be 1 for $v \in L$ and 0 for $v \notin L$. M than is said to accept L if $p(M(v) = 1) > 0.5$ for $v \in L$. M is said to decide L if for $v \in L$ we have $p(M(v) = 1) > 0.5$ and for $v \notin L$ we have $p(M(v) = 0) > 0.5$. In the class of problems solved by a PTM let us now differentiate between two alternative hypotheses H_0 and H_1. Let the hypotheses be defined as $H_0 : v \in L$ and $H_1 : v \notin L$. If M is considered to be a test which decides between $v \in L$ and $v \notin L$ with α as error of the first kind then $\alpha = p(M(v) = 0 \mid v \in L)$. If $1 - \beta$ denotes the error of the second kind, we get $1 - \beta = p(M(v) = 1 \mid v \notin L)$ where β is called the *power* of the test.

We are able to define classes of probabilistic algorithms according to their precision assuming now and furtheron that M is a PTM solving its problem in *polynomial time* [1]:

$$\begin{aligned} PP &= \{L \subseteq \{0,1\}^* \mid \exists M \in PTM \; and \; \alpha < 0.5\} \\ BPP &= \{L \subseteq \{0,1\}^* \mid \exists M \in PTM \; and \; \alpha < \varepsilon \; and \; 1 - \beta < \varepsilon \; and \; \varepsilon < 0.5\} \\ VPP &= \{L \subseteq \{0,1\}^* \mid \exists M \in PTM \; and \; \alpha < 0.5 \; and \; 1 - \beta = 0\} \end{aligned}$$

Despite the fact that there seems to be little difference between PP and BPP, this difference turns out to be of considerable importance in a more detailed investigation (see [Welsh]). Only in BPP and its subclasses the error of probabilistic algorithms can be made arbitrarily small by repeated tests. Obviously if $M \in PP$ M recognizes in polynomial time with error probability < 0.5, whereas if $M \in BPP$ M recognizes in polynomial time with error probability bounded by a constant ϵ less than 0.5. If $M \in VPP$ M recognizes in polynomial *average* time with zero error probability for inputs not in the language L. The classes BPP and VPP are usually considered as the class of *Monte Carlo* algorithms. Another class is ZPP. ZPP recognizes languages in polynomially bounded average run time and zero error probability:

$$\begin{aligned} ZPP = \{L \subseteq \{0,1\}^* \mid & \exists M \in PTM \; and \; (\alpha < 0.5 \; and \; 1 - \beta = 0) \\ & and \; \exists \; \overline{M} \in PTM \; for \; \overline{L} \; with \; \alpha' < 0.5 \; and \; 1 - \beta' = 0)\} \end{aligned}$$

where $\alpha' = p(\overline{M}(v) = 0 \mid v \in \overline{L})$ and $1 - \beta' = p(\overline{M}(v) = 1 \mid v \notin \overline{L})$, $\overline{L}$ is the complement of L in $\{0,1\}^*$ and $\overline{M}$ is the machine which accepts $\overline{L}$. Algorithms in ZPP are usually called *Las Vegas* algorithms.

The reader should note that both $M(v) = 1$ and $\overline{M}(v) = 1$ are possible – a situation which is called *indifference*. Let the probability of indifference be γ. γ can be reduced to arbitrary $\varepsilon > 0$ by repeating the tests. This class of algorithms with zero-sided errors consists of two algorithms with one-sided errors each. However, there is still some probability of not successfully solving the problem. We know that ([Gill])

$$\begin{array}{ccc} & \subseteq \text{BPP} \subseteq & \\ \text{P} \subseteq \text{ZPP} \subseteq \text{VPP} & & \text{PP} \subseteq \text{PSPACE} \\ & \subseteq \text{NP} \subseteq & \end{array}$$

is valid. (P, NP and $PSPACE$ are well known complexity classes explained for example in the book of [Garey, Johnson].)

---

[1] If there is a polynomial p such that M for every possible input v halts with at most $p(\mid v \mid)$ computing steps we consider M to work in polynomial time or the PTM is polynomially bounded. As run time Blum run time or average run time can be used. If nothing is mentioned, Blum run time is used.

Considering now the class of probabilistic algorithms randomized algorithms are a specialized concept of the former solving usually problems in P with no error and always. This need not be true for probabilistic algorithms in general [2]. A further example of a probabilistic algorithm not always solving the problem is [Babai]'s graph isomorphism testing algorithm. Note that there are also non-probabilistic algorithms solving a problem (asymptotically optimally) only with some error probability, for example Karp's algorithm for solving the Euclidean travelling salesman problem in the plane. In the following contribution we concentrate on using statistical and probability theoretical methods to accelerate algorithms solving deterministically by randomization. We use a search problem in P. (Other types of problems are given by Freivalds, Gill and others (see [Welsh], [Janko 1982, 1983])).

2 Adaptive Randomized Algorithms

2.1 Introduction

Let A denote the set of pure algorithms: $A = \{a_1, a_2, \ldots, a_m\}$. Let $D = \{d_1, d_2, \ldots, d_n\}$ denote the set of data instances of a problem π_k of size k belonging to the class of problems Π. Considering now the cost $c(a_i, d_j)$ (abbreviated c_{ij}) of running a_i with data instance d_j to solve π_k we restrict our considerations for reasons of simplicity on the run time as the only measure of cost. A *randomized* algorithm is then characterized by a distribution q on A. Similarly there is usually a prior distribution f on D which is known or assumed to be known. In this case statistical and economic decision theory is able to deliver several results for algorithm theory. Let us call $\sum_{i=1}^{m} q_i c_{ij}$ the expected *randomized* complexity of the algorithm characterized by q and $\sum_{j=1}^{n} f_j c_{ij}$ the expected *distributional* complexity of algorithm a_i. Statistical decision theory now tells us:

There is a pure algorithm which is at least as good as any randomized algorithm in its distributional complexity (see e.g. [Ferschl], pp 89).

As we will learn in the following this need not help us if we are not able to identify this algorithm with little cost. Now let us consider the worst case of the pure algorithms under each data instance $\phi(a_i) = max\{c_{ij} \mid j = 1, 2, \ldots, n\}$. Choosing the best pure algorithm with respect to the worst case we choose the algorithm a^* which gives the minimum of the set $\{\phi(a_i) \mid i = 1, 2, \ldots, m\}$; that means $a^* = \inf_{\{i\}} \max_{\{j\}} c_{ij}$. a^* is called the minimax-algorithm. We are now ready for the next result:

There is eventually a randomized algorithm which is in its worst case better than any pure minimax-algorithm. This is the equalizer algorithm, an algorithm for which we get $\sum_{i=1}^{m} q_i c_{ij}$ is equal for each value of j (see [Ferschl], pp 85 and [Ferguson] for more details).

However the fundamental theorem of two-person zero-sum game theory gives us the important equality

$$\inf_{q} \max_{\{j\}} \sum_{\{i\}} q_i c_{ij} = \sup_{f} \min_{\{i\}} \sum_{\{j\}} f_j c_{ij}$$

(see [Ferguson], p 57). In other words the theorem above tells us that there is always a distribution f on the set of data instances which makes the computation at least as costly as the best randomized algorithm - characterised by q - we can choose using min-max selection for nature and the decision maker. Further interesting results can be derived with respect to algorithm theory from the close connection of decision and algorithm theory (see [Yao 1977b]).

Previous works have dealt with the case of randomized search algorithms on linear data sets and orthogonal data sets ([Janko 1985b], [Janko 1980], [Janko 1973], [Bentley, Stanat and Steele], [Leighton and Lepley]) proving that by randomization search effort can be reduced from $O(n)$ to $O(n^{0.5})$ (linear data set) and from $O(n^{0.5})$ to $O(n^{\frac{1}{3}})$ (quadratic orthogonal list). This paper is concerned with the problem of randomized search in the case of an unkown distribution of sequential search steps drawing sequential search algorithms.

In this case - for various possible reasons - the situation changes. There are many variations of the problem of optimal stopping and randomized search which can be applied here. However, the two types of randomized algorithms presented proved to be practically the most useful (see [Janko 1976a], [Janko 1976b], [Janko 1976c], [Janko 1977]). There are at least two possible ways of solving this problem:

---

[2] The reader should note that this terminology differs from the terminology used by some authors (see for example [Welsh]). Also Las Vegas and Monte Carlo algorithms are expressions which are not consistently used.

1. **In addition to the search phase with search costs c_s we introduce a testing phase to check the quality of the randomly drawn algorithm (in terms of search steps) at cost c_T and to stop or continue search with respect to the test result (see [Janko 1985d], [Hartmann 1985], [Guenther], [MacQueen]).**
2. **We adapt the multinomial distribution of chances in random search by updating the probabilities $q(j)$ dynamically according to the experience in the search (see [De Groot]).**

Introducing one or more testing phases means that we have to know the 3-dimensional or higher--dimensional distribution of the correlated random variables for the success of searching, testing and the true but unknown real value of sequential search efforts. Whereas the principal method has been derived (see [Hartmann 1985], [Janko 1985d]), there is to the authors' knowledge no experience with its application. Stopping with multinomially distributed rewards and simultaneous Bayes adaption of the distribution function in algorithm search was first proposed by [Janko 1976a] using the method of [Randolph].

2.2 Optimal Randomized Search with Bayes Adaptation

It is well known that the family of Dirichlet distributions is a conjugate family for observations which follow a multinomial distribution (see [De Groot], [Randolph]). It must be emphasized that using the multinomial distribution we assume prior knowledge of the maximum value of observable algorithms. Let the prior density function of the multinomial probabilities for

$$x = (x_0, x_1, \ldots, x_{k-1})$$

be equal to

$$f(x) = \frac{\Gamma(\alpha_0 + \cdots + \alpha_{k-1})}{\Gamma(\alpha_0) \ldots \Gamma(\alpha_{k-1})} x_0^{\alpha_0 - 1} \cdots x_{k-1}^{\alpha_{k-1}-1}$$

for $x_i > 0$, $\alpha_i > 0$ for $i = 0, 1, \ldots, k-1$, $\sum_{i=0}^{k-1} x_i = 1$ and $\alpha = \sum_{i=0}^{k-1} \alpha_i$.

The posterior distribution function after making n oberservations and letting n_j be the number of observations of j and $n = \sum_{j=0}^{k-1} n_j$ is

$$f_n(x) = \frac{\Gamma(\alpha_0 + \cdots + \alpha_{k-1} + n)}{\Gamma(\alpha_0 + n_0) \ldots \Gamma(\alpha_{k-1} + n_{k-1})} x_0^{\alpha_0 + n_0 - 1} \cdots x_{k-1}^{\alpha_{k-1} + n_{k-1} - 1}.$$

The multinomial probability that the $(n+1)^{-th}$ observation is equal to j is

$$q^n(j) = \frac{\alpha_j + n_j}{\alpha + n} \qquad j \in \{0, 1, 2, \ldots, k-1\}.$$

Using the tuple $(\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-1})$ we can determine the probability vector $(q_0, q_1, \cdots, q_{k-1})$ but not vice versa. The value of α plays the most important role in the use of our prior distribution; it can be interpreted as the *degree of belief* we are having in our prior distribution. Assuming α to be very large means that observations have little impact on the value of our multinomial probabilities. With $\alpha \to \infty$ new observations would not change our prior probabilities at all. Low values of α, on the contrary, make our prior multinomial probabilities very vulnerable to experience. Now let v_n be equal to $x_n + cn$ in the case of no recall and equal to $min(x_1, x_2, \ldots, x_n) + cn$ in the case of recall. Let the prior density before we observe the $(n+1)^{-th}$ alternative be $f_n(x)$. Then the multinomial prior probabilities are given by $q^n(j)$. The expected value from an additional observation is equal to

$$T_n(v_n) = \sum_{j=0}^{\lfloor v_n \rfloor} (v_n - j) q^n(j).$$

We would therefore stop when $T_n(v_n) \leq c$ and continue to search when $T_n(v_n) > c$. Using

$$T_n(v_n) = \sum_{j=0}^{\lfloor v_n \rfloor} (v_n - j) q^n(j) = (\alpha + n)^{-1} \sum_{j=0}^{\lfloor v_n \rfloor} (v_n - j)(\alpha_j + n_j)$$

and bearing in mind that

$$\alpha_j = \alpha q_0(j)$$

we get

$$T_n(v_n) = (\alpha + n)^{-1}\alpha T(v_n) + (\alpha + n)^{-1}\sum_{j=0}^{\lfloor v_n \rfloor}(v_n - j)n_j.$$

$T(v_n)$ equals

$$\sum_{j=0}^{\lfloor v_n \rfloor}(v_n - j)q_0(j),$$

which is the expected value of an additional observation assuming a known multinomial distribution function with the probabilities $q_0(j)$. In sampling with recall, v_n is the best algorithm observed in all observations, the values of n_j are therefore zero for $j = 0, 1, \ldots, \lfloor v_n \rfloor$. That means we get for the case of recall

$$T_n(v_n) = (\alpha + n)^{-1}\alpha T(v_n)$$

and for the case of no recall

$$T'_n(v_n) = (\alpha + n)^{-1}\alpha T(v_n) + (\alpha + n)^{-1}\sum_{j=0}^{\lfloor v_n \rfloor}(v_n - j)n_j.$$

As $\lim_{\alpha\to\infty} T_{n+1}(v_n) = T(v_n)$ and $\lim_{\alpha\to\infty} T'_{n+1}(v_n) = T(v_n)$, we can see that the case of fixed prior probabilities based on an infinite coefficient of belief α is a very special case of the adaptive formulation. We shall also consider the problem of the influence of α. With increasing α and observations with recall the value of $T_n(v_n)$ increases with α. Or to put it differently: if two searchers have the same prior distribution and have observed the same algorithms, the one with a higher degree of belief α in his prior probabilities continues random search while the one with the lower degree of (subjective) belief in his prior probabilities has already stopped the random search. This statement can be made more precise if we consider the discrete uniform distribution: the maximum number of observations is then the solution of

$$k - 1 = -0.5 + \sqrt{0.25 + 2c + \frac{2nkc}{\alpha}}$$

or

$$k^2 - k\left(\frac{\alpha + 2nc}{\alpha}\right) - 2c = 0.$$

Solving this equation for n we get for the maximum value n^+ of n:

$$n^+ = \frac{\alpha(k-1)}{2c} - \frac{\alpha}{k}.$$

The term dominating this expression is $\alpha(k-1)/2c$. Thus the number of algorithms searched for will be primarily dependent on the degree of belief in our prior distribution (expressed through α), the value of k and the search costs. The higher the search costs, the lower the maximum number of observations. The larger k, the larger is the maximum number of observations and the larger is the variance. It is seldom mentioned that with this search strategy k has to be fixed in advance. As k has to be fixed in advance and is *not* revised by the adaptive strategy, this model of search will rather show a tendency of making the maximum number of observations larger than necessary. This is due to the fact that k has to be chosen such that it covers also the most unfortunate cases which eventually do not even occur. The basic assumption in this adaptive strategy is that we want to converge to one true distribution which is unknown. But this is not the case in many applications. In fact, wherever the true unknown underlying distribution changes over time, the strategy described leads to wrong results. Simulation experiments have shown that the prior distribution converges relatively fast to the true distribution. However, as mentioned earlier, since the true distribution would itself be rather a function of the underlying environment and other factors, the true distribution itself will vary with time. But the larger the coefficient of confidence α, the more slowly will the distribution converge with the true distribution. With $\alpha + n \to \infty$ the *prior* will obviously not react any more to a change of the true distribution. What therefore should be done in such a case is to start to reduce $\alpha + n$ by the initial observations if $\alpha + n \geq g(\alpha)$ for some $g(\alpha)$. $g(\alpha)$ has to be fixed individually according to the amount and the rate of the change of the underlying true distribution with respect to the rate of the observations done in the same time interval. g should also be a function of the confidence coefficient α (as this coefficient determines the speed of adaption).

2.3 The Use of Adaptive Models with Randomized Search

If we have equidistant key sets, we can directly learn from the value of the distance function for each algorithm drawn what its sequential search effort would be in terms of sequential search steps. Thus we are able to update the multinomial distribution of chances of random search during the search. [Gasber] made two types of experiments: in the first one he used partial knowledge of the distribution of random search; in the second experiment he used not fully occupied orthogonal lists together with an adaptation of the distribution of sequential search effort to determine an "average good" stopping interval ($= [0, v]$). An average good stopping interval is necessary for reasons of simplicity as each key searched for has its own distribution of chances in random search which very often will be different from one another. Therefore, for each key searched for an optimal stopping interval would have to be determined for each random search step. An average good stopping interval is an interval for all distributions during the entire search process. Two simplifications were used in the experiments:

- The stopping intervall is determined at the beginning of one search problem and not updated according to the new experiences until the key searched for is found, and the problem is solved.
- The stopping interval $[0, v']$ is the only one used during one search process, and no revision of the stopping interval during the search process was realized as would be necessary using optimal Bayes stopping.

The latter type of simplification is also used for the first type of experiments.

2.4 Learning the Shape of Non-quadratic Orthogonal Lists for Optimal Random Search

In the first type of experiments the stopping interval for random search is determined using first a uniform distribution over all possible values of sequential search steps. Determining on this basis the optimal stopping interval $[0, v']$ simply by $v' = \lceil \sqrt{2 \times n_1 \times n_2 \times \ldots \times n_k} \rceil$ where the total number of list positions is $N = \prod_{j=1}^{k} n_j$, the distribution was adapted during random search as described above - the stopping interval was not (i_j determines the number of rows of the orthogonal list in the j-th dimension). After a successful solution of a search problem a new search problem is started with a recalculated stopping interval $[0, v']$ using the revised multinomial distribution. The simulation was based on 20 repetitions. The minimum, the average and the maximum number of random search accesses until stopping is counted. In addition, the number of search problems posed until the randomized search has learned the optimal stopping interval is measured:

exp.	shape of list	no. of	initial	optimal	no. of random accesses			no. of search problems		
no.		keys	value of v'		min	ave	max	min	ave	max
1	16x32	512	10	15	450	733	892	84	151	196
2	4x8x16	512	8	11	188	256	394	38	81	137
3	22x45	990	12	19	—	> 5000	—	—	—	—
4	5x10x20	1000	9	13	378	586	791	67	175	219
5	2x4x8x16	1024	8	12	598	1278	1800	177	431	608
6	32x64	2048	14	24	—	> 5000	—	—	—	—
7	6x13x26	2028	10	15	358	503	799	57	113	178
8	3x4x9x19	2052	8	13	190	297	400	36	89	129

In the experiments nos 3 and 6 the number of random accesses needed was rather large. Therefore the experiment was repeated with triangular shaped prior distributions. The number of random accesses is reduced dramatically by this improved initial guess to an average of 332 at experiment no 3 and to an average of 96 with experiment no 6.

2.5 Learning an Average Good Stopping Interval in Sparsely Occupied Orthogonal Data Sets

If we admit that not every cell of the orthogonal list is occupied by an element, we get a sparsely occupied orthogonal list. Searching in such a sparsely occupied orthogonal list offers a new variety of problems. First, all orthogonal list elements eventually have their own optimal stopping interval. Second, to be able to reach every orthogonal list element with certainty with a randomly drawn sequential search algorithm, it might be necessary to introduce dummy orthogonal list elements without data just to have the necessary linkage to reach an element. In an $n_1 \times n_2 \times \cdots \times n_k$ dimensional orthogonal list we take the $k-1$ dimensional subspace with the least number of non-occupied elements and fill these with dummy elements to be able to reach every orthogonal list element.

Randomized sequential search in non-fully occupied orthogonal lists opens up a new variety of problems:

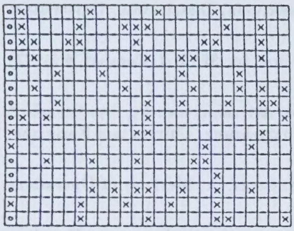
1. For each algorithm used for sequential search we have a distribution of sequential search cost which depends on the rock of the sequential search.
2. Using the distance function value to estimate the number of sequential search steps in sequential search is not necessarily accurate even if the key set is equidistant.
3. The sequential search cost of the randomly accessed sequential search algorithms can only roughly be estimated and therefore the distribution function of the chances in algorithm search can only roughly be adapted by Bayesian methods. To derive an optimal stopping interval will therefore not be possible in most cases. Just an *approximately good* stopping interval will be derivable.

Despite these problems simulation experiments have been encouraging for low occupancy rates. Using Bayesian learning and approximate stopping intervals with low to medium occupancy rates assuming uniformly distributed orthogonal list elements in the orthogonal list structures, simulation experiments were executed by [Gasber]. Results for the following cases were given:

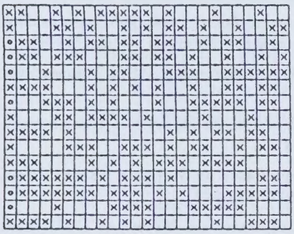
- A: Using an *average good* distribution of the cost of executing randomly accessed sequential search algorithms.
- B: Using for each key searched for the exact distribution of chances accessing the sequential search algorithms randomly.
- C: Using the distribution of chances of random algorithm search under the assumption of a fully occupied orthogonal list (whereas in fact we have not full occupancy).

For each of these cases the optimal stopping interval value (under simplifications), the average number of sequential accesses in sequential search, the average number of random search steps and their sum is given. The simulation experiments are carried out for two occupancy densities: 20 and 50 percent of the orthogonal list positions are occupied by real keys. The results indicate that almost fully occupied orthogonal lists are best used as if they are at least moderately occupied and the optimal stopping interval is derived from this consideration. Whereas having only a 20 percent occupancy rate, the usage of an *average good* distribution of chances in random search is successful. However, we should not forget that a uniformly distributed set of orthogonal list elements was used. With a more degenerate distribution of the list elements in the orthogonal list structure the results will eventually deviate severely.

Abbreviations used in the following tables:
$\times$:= true orthogonal list element
o := dummy element
$\sqcup$:= no element



Occupancy Rate 20 %	*A*	*B*	*C*
optimal stopping interval value *v*	9	7 . . . 10	12
average number of sequential search steps	3, 721	3, 566	3, 879
average number of random search steps	6, 508	5, 413	8, 104
total number of average search steps	10, 229	8, 979	11, 983



Occupancy Rate 50 %	*A*	*B*	*C*
optimal stopping interval value *v*	13	9 . . . 14	12
average number of sequential search steps	5, 886	5, 284	4, 331
average number of random search steps	8, 200	6, 414	7, 986
total number of average search steps	14, 086	11, 698	12, 317

3 Conclusion

The authors have given an overview of a special class of adaptive randomized search algorithms. We have tried to indicate the close similarity between problems in economic decision theory (state preference model), statistical decision theory and algorithm theory with respect to the computational effort. There is a general tendency in the automatic information processing market to move from hierarchies to markets as problem solutions and data representations become more portable. This follows from the increasing acceptance of representation standards and the broad acceptance of increasingly producer independent operating systems like MS-DOS and UNIX and their variants. From the point of risk evaluation in management and economics, the economics of automated problem solving and its implications to the design of such solutions and their implemenation are hardly investigated. In addition the combination of decision theory and algorithm design methods seems to offer further research opportunities.

References

[Babai] Babai, L., Monte-Carlo Algorithms In Graph Isomorphism Testing, Working Paper, D.M.S. No. 79-10, CP.6128, SUCC. "A", Montreal Quebec H3C 3J7, August (1979)

[Bentley, Stanat and Steele] Bentley, J.L., Stanat, D.F., Steele, J.M.: Analysis of a Randomized Data Structure for Representing Ordered Sets, 19th Annual Allerton Conf. on Comm., Control and Comp., Allerton House, Monticello (1981) 364-373.

[De Groot] De Groot, M.H., Optimal Statistical Decisions, McGraw Hill, New York (1970).

[Ferguson] Ferguson, J., Statistical Decision Theory, Academic Press (1967)

[Ferschl] Ferschl,F., Nutzen- und Entscheidungstheorie, Westdeutscher Verlag, Opladen (1975)

[Garey, Johnson] Garey, M.R., Johnson D.S., Computers and Intractability, Freeman, San Francisco (1979)

[Gasber] Gasber, R., Untersuchung von adaptiven randomisierten Datenstrukturen, Dissertation, Universitaet Karlsruhe, Karlsruhe (1984).

[Gill] Gill III.,J.T., Computational Complexity of Probabilistic Turing Machines, in: SIAM J. Comput. 6 (1977) 675-694.

[Guenther] Guenther, O., Die Normalverteilung im MacQueen-Modell zur Alternativensuche, Studienarbeit am Institut fuer Angewandte Betriebswirtschaftslehre und Unternehmensfuehrung, Universitaet Karlsruhe, Karlsruhe (1984).

[Hartmann 1984] Hartmann, J., Searching and Testing with Multinomial Rewards, in: Methods of Operations Research 52, Henn, Kall et al. (eds.), Athenaeum, Koenigstein (1984).

[Hartmann 1985] Hartmann, J., Wirtschaftliche Alternativensuche mit Informationsbeschaffung unter Unsicherheit, Dissertation, Universitaet Karlsruhe, Karlsruhe (1985).

[Janko 1973] Janko, W.H., Optimale Stichprobensuchverfahren bei Implementierung von Datenbestaenden mittels orthogonaler Listen.
Teil I: Die Implementierung von Datenbestaenden mittels orthogonaler Listen.
Teil II: Optimale Stichprobensuche in orthogonalen Listen, Angewandte Informatik 11,12/73 (1973).

[Janko 1976a] Janko, W.H., Stochastische Modelle in Such- und Sortierprozessen, Forschungsergebnisse aus dem Revisionswesen und der betriebswirtschaftlichen Steuerlehre, Bd.3, Duncker u. Humbolt, Berlin (1976).

[Janko 1976b] Janko, W.H., A List Insertion Sort for Keys with Arbitrary Key Distribution, ACM Transactions on Mathematical Software, Vol. 2, No. 2, 204-206 (1976)

[Janko 1976c] Janko, W.H., ALGORITHM 505: A List Insertion Sort for Keys with Arbitrary Key Distribution, ACM Transactions on Mathematical Software, Vol. 2, No. 2, 204 (1976)

[Janko 1977] Janko, W.H., An Insertion Sort for Uniformly Distributed Keys Based on Stopping Theory, Proc. Int. Comput. Symp., Morlet E., Ribbens D. (Eds.), North Holland (1977) 373-379.

[Janko 1980] Janko, W.H., Probabilistic Algorithms and the Efficient Use of Statistical Models of Decision in their Design, in: Methods of Operations Research, Henn et al. (eds.), vol.36, Athenaeum, Koenigstein (1980).

[Janko 1982, 1983] Janko, W.H., A Bibliography of Randomized Algorithms, EATCS Bulletin, No. 18 (1982), Addendum, No. 20 (1983)

[Janko 1985a] Janko, W.H., Searching for the Best Offer When the Distribution of Offers is Truly Unknown, in: Contributions to Econometrics and Statistics Today, Schneeweiss und Strecker (eds.), Springer, Berlin/Heidelberg (1985).

[Janko 1985b] Janko, W.H., Randomized Search in Orthogonal Lists and on Peripheral Storage Devices, 23rd Annual Allerton Conference on Communication, Control and Computing, Monticello (1985), 850-858

[Janko 1985c] Janko, W.H., On the Efficiency of a Model of Search for the Best Offer (with some Knowledge of the Underlying Distribution) and its Assumptions, in: Guthardt, H., Henn, R., Kremer, A., Pallaschke, D. (eds.), Aspekte bankwirtschaftlicher Forschung und Praxis, Frankfurt (1985).

[Janko 1985d] Janko, W.H., Hartmann, J., Flexible Informationsbeschaffung in Alternativensuchproblemen, in: Ballwieser, W., Berger, K.-H.(Eds.), Information und Wirtschaftlichkeit, Hannover (1985).

[Leighton and Lepley] Leighton, T., Lepley, M.: Probabilistic Searching in Sorted Linked Lists, 20th Allerton Conf. on Comm., Control and Comput., Monticello (1982) 500-506.

[MacQueen] MacQueen, J.B., Optimal Policies for a Class of Search and Evaluation Problems, Management Science, vol.10, no.4 (1964).

[Rabin] Rabin, M.O., Probabilistic Algorithms, in: Traub, J.F., (ed.), Algorithms and Complexity (Academic Press, New York, 1976), 21-39.

[Randolph] Randolph, P.H., Optimal Stopping Rules for Multinomial Observations, Metrika, vol.16 (1968).

[Solovay and Strassen] Solovay, R., Strassen, V., A Fast Monte-Carlo Test for Primality, SIAM Journal of Computing, 6 (1977) 84-85.

[Welsh] Welsh, D.J.A., Randomized Algorithms, Discrete Applied Mathematics 5 (1983), 133-145.

[Yao 1977a] Yao, A.C., A Lower Bound To Palindrome Recognition By Probabilistic Turing Machines, Technical Reports STAN-CS-77-647, Stanford University (1977).

[Yao 1977b] Yao, A.C., Probabilistic Computations: Toward a Unified Measure of Complexity, 18th Annual Symposium on Foundations of Computer Science (1977).

THE SET UNION PROBLEM WITH DYNAMIC WEIGHTED BACKTRACKING

GIORGIO GAMBOSI [1]
Istituto di Analisi dei Sistemi ed Informatica - C.N.R., Roma, Italy
GIUSEPPE F. ITALIANO [2]
Department of Computer Science, Columbia University, New York, NY
MAURIZIO TALAMO [1]
Dipartimento di Informatica ed Applicazioni, Università di Salerno, Salerno, Italy

ABSTRACT

In this paper, an extension of the set union problem is considered, in which dynamic weighted backtracking over sequences of unions is permitted. That is, we associate to each union a real number, called its *weight*, subject to dynamic updates (both increases and decreases). Furthermore, we allow to backtrack to the state of the structure just before either any union operation or the union of maximal weight thus far performed. We describe a data structure which supports each of the above operations in *O(lgn)* worst-case time with an *O(n)* space usage, where n is the total number of items to be maintained. We prove that this bound is tight for a general class of algorithms previously defined in the literature. Furthermore, we show also how a refinement of this data structure can achieve better amortized bounds. Motivations for studying this problem arise in the introduction of heuristics in Prolog interpreters.

1. INTRODUCTION

Several extension to the classical set union problem [**1, 3, 4, 9, 11, 13, 18, 21, 22, 24, 25**] have been recently introduced in Theoretical Computer Science.

Mannila and Ukkonen [**14**] proposed a variant of this problem, by introducing a new operation defined as follows.

deunion : undo the last union performed, i.e. return to the state immediately preceding the execution of such union.

Westbrook and Tarjan [**16, 27**], derived a $\Theta(lgn/lglgn)$ upper and lower bound on the amortized time complexity of the problem using a data structure with an $O(n)$ space complexity.

---

[1] Supported in part by the ALPES Esprit Project.

[2] Supported in part by an IBM Graduate Fellowship. On leave from Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy.

Gambosi, Italiano and Talamo [**7, 8**] considered a generalization of such problem to the case where a (real) weight is associated to each union and the deunion operation is substituted by *backtrack* : return to the state immediately before the execution of the union of largest weight thus far performed.

In the following, we will denote such operation as $backtrack_2$.

Motivations for the study of this extension of the set union problem arise from the implementation of search heuristics in the framework of Prolog environment design [**10, 26**]. In such a context, a sequence of union operations models a (sequence of) unification(s) between terms [**15**], while the weight associated to a union allows to evaluate the "goodness" of the state resulting by the unification to which the union is associated. Thus, backtracking corresponds to return to the "most promising" state examined so far, in the case of a failure of the current path of search.

The data structure presented in [**8**] supports each union in $O(lglgn)$ worst-case time, each find in $O(lgn)$ worst-case time and each backtrack in $O(1)$ worst-case time, with a total of $O(n)$ space usage.

In this paper, we extend the set of operations specified above by allowing to update (both to increase and to decrease) the weight associated to each union already performed. This operation adds more heuristic power to the algorithms for Prolog interpreter memory management, and henceforth improves the practical performance of the previous known blind uniform-cost algorithms.

Moreover, the possibility to backtrack to the state just before any union performed so far is introduced: in the following, we will refer to this operation as $backtrack_2$. This makes it possible to implement several hybrid strategies based on best-first search combined with backtracking [**12, 20**] in the framework of the control of the resolution process.

In particular, we present data structures which support the following operations.

$union(X, Y, Z, w)$: combine sets X, Y into Z and assign weight w to the operation.
$find(v)$: return the name of the set containing element v.
$increase\_weight(i, \alpha)$: increase by α the weight of the i-th union performed.
$decrease\_weight(i, \alpha)$: decrease by α the weight of the i-th union performed.
$backtrack_1$: undo all the unions performed after the one with maximal weight.
$backtrack_2(i)$: undo all the unions performed after the i-th one.

The main results of the paper can be summarized as the following table shows.

Operation	**Worst-case per operation**	**Amortized**
$UNION(X,Y,Z,w)$	$O(lgn)$	$O(lgn)$
$FIND(v)$	$O(lgn)$	$O(lgn)$
$DECREASE\_WEIGHT(i,\alpha)$	$O(lgn)$	$O(lgn)$
$INCREASE\_WEIGHT(i,\alpha)$	$O(lgn)$	$O(1)$
$BACKTRACK_1$	$O(lgn)$	$O(1)$
$BACKTRACK_2(i)$	$O(lgn)$	$O(1)$

We prove also that no better worst-case bound per operation can be achieved for a rather general class of algorithms previously defined in the literature, known as nonseparable pointer algorithms [**17, 18, 27**].

In [**6**] a partially persistent data structure for set union is introduced which makes it possible to perform unions in $O(1)$ and finds in the past in $O(lgn)$ time per operation. This result can be used, together with the ones derived in this paper, to obtain a data structure which supports unions in $O(1)$ time, finds in the past in $O(lgn)$ time and allows to return to any arbitrarily chosen previous version of the data structure in $O(logn)$ amortized time.

The remainder of the paper consists of five sections. In section 2 a new data structure, called *Backtracking Queue*, is introduced. This data structure will be used in section 3 to establish the worst-case per operation upper bound. In section 4, we will show also that this bound is tight for a general class of algorithms, while in section 5 better amortized bounds are derived. Section 6 contains some concluding remarks.

2. BACKTRACKING QUEUES

In this section we will introduce a new data structure, called *Backtracking Queue* (in short BQ), which maintains items to which real values are associated. Each time a new item x enters a BQ, an ordinal number $p(x)$ is associated to it, referred to as its position into the queue. The operations defined on BQ's are the following.

$insert(x, v, Q)$: insert item x with value v into the BQ Q. If exactly i items were previously in Q, $p(x) = i + 1$.

$increase(i, d, Q)$: if Q contains fewer than i elements, then return Q. Otherwise, increase by $d > 0$ (d real) the value associated to the item x such that $p(x) = i$.

$decrease(i, d, Q)$: if Q contains fewer than i elements, then return Q. Otherwise, decrease by $d > 0$ (d real) the value associated to the item x such that $p(x) = i$.

$back_1(Q)$: return to the state of the BQ Q just before the current item of largest value was inserted into Q.

$back_2(i, Q)$: if Q contains fewer than i elements, then return Q. Otherwise return to the state of the BQ just before the item currently in position i was inserted.

In other terms, BQ's are priority queues for which some sort of getting back to the past is allowed.

BQ's with n elements may be represented by means of a particular class of binary trees (referred to as BQ trees), which are defined as follows. Each n-item BQ tree has the following structure:

(i) at level 0 there are exactly n nodes (leaves);

(ii) at level $k > 0$ there are exactly $l_k = \lceil \frac{l_{k-1}}{2} \rceil$ internal nodes and no leaves.

Furthermore, all internal nodes at level k ($k > 0$), with the only exception of the rightmost one, have two children.

In such an implementation, leaves represent items $\{x_1, x_2, \ldots, x_n\}$ in the BQ, organized in a left-to-right order according to their positions $\{p(x_1), p(x_2), \ldots, p(x_n)\}$ in the queue itself. Thus, the leaf of rank k in the ordering stores the value associated to the item x with $p(x) = k$, the value $p(x)$ and a reference to the item itself. Each internal node stores a value equal to the maximum between the values stored in its children.

Figure 1 shows such types of trees obtained for $n = 1, \ldots, 5$.

The leaves of a BQ are accessible by means of pointers organized in a balanced tree [**1**]. More precisely, the generic item in such a balanced tree has a key i, $1 \leq i \leq n$, and stores a pointer to the i-th leaf from the left (and henceforth the item in position i) in the BQ. In the following, we assume that a procedure ACCESS(i) searches for the key i in the balanced tree and returns a pointer to the i-th leaf from the left in the BQ. Balanced trees can supported insertion, deletion and search of items in a worst-case time which is logarithmic in their size [**1**] and hence the cost of accessing each leaf of a BQ will be $O(lgn)$. Moreover, we will denote as i_{max} the size of the BQ.

We define, as usual, the height $h(T)$ of a tree T to be the maximum number of nodes on a path from the root of T to a leaf and the size $s(T)$ of T as the number of nodes in T. The following lemma then holds.

LEMMA 2.1: Let T_n be an n-item BQ tree. Then $h(T_n) \leq \lceil lg_2 n \rceil + 1$ and $s(T_n) = O(n)$.

PROOF: Trivial •

Let us now turn to the implementation of all the operations defined on a BQ.

When an *insert*(x, v, Q) has to be performed, first of all item y in position i_{max} is accessed. Then, starting from that leaf, the tree is ascended until the first node with only one child, referred to as $apex(y)$, is entered. If no $apex(y)$ is encountered, then a new node is created and we make the $root(T_n)$ left child of such new node, thus increasing by one the height of the tree: this new node is now $apex(y)$ by definition.

Let us denote as r the level of $apex(y)$ (Fig.2). A new leaf containing item x with key v is created and pointed to by ACCESS($i_{max} + 1$). A chain of length r, with all internal nodes labeled with v, is introduced from such leaf to $apex(y)$ in such a way that it is the leftmost leaf in the right subtree of $apex(y)$. If $r < h(T_n)$, then a number of values in the internal nodes of the path from $apex(y)$ to the root are updated by letting the value v ascend as much as possible. Finally, i_{max} is incremented by one.

When dealing with an *increase*(i, d, Q) [*decrease*(i, d, Q)] operation, first of all i and i_{max} are compared. If i is larger than i_{max}, then no operation is required. Otherwise, the item y in the i-th position is accessed and the real number d is added [borrowed] to its key. Also, the values in the internal nodes of the path from y to the root are properly updated.

A $back_1(Q)$ may be implemented by performing a top-down splitting starting from the root and following the labels of the maximum key in the BQ. If j is the position of the item of largest key (which may be read in the leaf reached at the end of splitting), i_{max} is set to $j - 1$.

Finally, if a $back_2(i, Q)$ has to be performed, a new BQ tree containing only the elements whose position is less than i must be returned. If i is larger than i_{max}, then nothing has to be done. Otherwise, we perform a bottom-up splitting starting from the leaf reached through ACCESS(i).

The time complexity of the preceding operations is analysed in the following theorem.

THEOREM 2.1: There exists an implementation of BQ's which supports each *insert*, *increase*, *decrease*, $back_1$ and $back_2$ operation in $O(lgn)$ worst-case time.

PROOF: The proof will be given in the full paper.•

The space complexity of the structure can be characterized by the following theorem:

THEOREM 2.2: There exists an implementation of $BQ's$ which supports each *insert*, *increase*, *decrease*, $back_1$ and $back_2$ in $O(lgn)$ worst-case time and requires $O(n)$ space.

PROOF: The proof will be given in the full paper.•

3. WORST-CASE PER OPERATION ANALYSIS

In this section we will introduce a data structure for the original problem of supporting the *union*, *find*, *increase_weight*, *decrease_weight*, $backtrack_1$ and $backtrack_2$ operations. The data structure supports each operation in $O(lgn)$ worst-case time and requires $O(n)$ space. We will show also that no better bound is possible for a general class of algorithms. We will only give an highlight of the data structure.

During the execution of any sequence of *union*, *find*, $backtrack_1$, $backtrack_2$, *increase_weight* and *decrease_weight*, all the items are partitioned into a collection of disjoint sets. In the following, we will refer to a *union* as *valid* if it has not been undone by backtracking and as *void* otherwise.

It is not difficult to see that at any time the actual partition is the same that would have been resulted from simply applying the currently valid unions to the initial set of singletons, in the exact order in which such unions were performed in the actual sequence of operations. This individuates a virtual sequence of valid unions. At any time it is henceforth possible to univocally denote each valid *union* by the ordinal number it gets in that virtual sequence of unions. Furthermore, it can be proved that each *union*, as long as it is valid, maintains the same ordinal number it was given at the time of its creation.

The ideas underlying the data structure which supports the above operations are the following.

As in most of the algorithms which deal with versions of the set union problem, we maintain every set as a tree, whose root contains the name of the set. When a *union* is performed, exactly one link between two nodes is introduced, which is associated to the *union* operation. Hence, also a link is said to be *valid* or *void* according to its corresponding *union*: valid links return a connection which has not yet been cancelled by backtracking.

Furthermore, to each *union* at the time of its creation is associated its rank in the virtual sequence of valid unions. This number is also stored into the link corresponding to that *union*. In our approach, links removed by backtracking are not removed immediately from the structure. As a consequence, both valid and void links may be in the data structure at the same time and henceforth we need to introduce some technique to discriminate between the two kinds of links.

In order to overcome the $O(n)$ time bound obtained for the *find* operation when no linking by size or by rank [**24**] is performed, the ranks of the nodes are maintained during the evolution of the structure. Following an idea previously described in [**7, 8**], to each node a balanced tree [**2, 19**], referred to as $rank(x)$, is associated. Each item of $rank(x)$ (in the sequel referred to as a *rank* of x) corresponds to a link l entering x. Its key in the balanced tree $rank(x)$ is the ordinal number of (the *union* which introduced) l, while it also stores the rank of x immediately after the introduction of l. Notice that, as a consequence of this definition, there exists a one to one correspondence between links and rank items: in order to allow fast references between ranks and links, a bidirectional pointer is introduced between a rank item and its corresponding link.

The notion of validity can now be extended to a rank of any node x: hence, a rank of x is said to be valid if and only if the corresponding link is valid, otherwise it is said to be void.

In order to restore the correct rank of any node x in presence of valid and void ranks, the largest valid rank (i.e. the value corresponding to the valid union with largest ordinal number) must be individuated in $rank(x)$. This information clearly turns out to be useful while implementing linking by rank.

Similarly, balanced trees may be also associated to each node x in order to store the names assigned to sets represented by trees rooted at x during the evolution of the structure: such a balanced tree will be referred to as $name(x)$.

As in the case of the $rank$ balanced trees, each item in $name(x)$ corresponds to a link l entering x, has associated the ordinal number of that link as a key and stores the name associated to x after the introduction of l. Once again, there are bidirectional pointers between names (items in $name(x)$) and the corresponding links and the notion of validity can be extended to the names (fig.3). When the current name of a root r must be individuated, a search for the largest valid item in $name(r)$ must be accomplished (i.e., a search for the name corresponding to the valid $union$ with largest ordinal number).

The weights of the valid unions are maintained in a Backtracking Queue Q subject to the garbage collection mechanism described in the previous section. More precisely, each item in Q represents a valid $union$ together with its current weight, while the position into Q is its ordinal number in the virtual sequence of valid unions. The implementation of BQ's previously given in the previous section is extended by letting each node of the balanced tree which allows to access the leaves of the BQ contain one extra bidirectional pointer to the link introduced by that $union$ (fig.4).

In order to accomplish efficiently the test for the validity of a link, the following invariant (referred to as *link consistency*) is maintained: "*If there are k valid unions at a given time, then a union present in the structure is void if and only if its corresponding link is labeled with an ordinal number larger than k*".

Notice that whenever link consistency holds, the validity of a link can be verified in constant time by simply comparing its label with the value i_{max} available in the Backtracking Queue.

The different operations can now be implemented as follows.

$union(X, Y, Z, w)$ - Let us denote by x and y the canonical elements of sets X and Y. Remove all links leaving x and y together with their associated ranks and names. By following the link pointer of ACCESS($i_{max} + 1$) locate the link (if any) previously introduced with ordinal number $i_{max} + 1$. Since this link has an ordinal number exceeding i_{max}, due to link consistency it is certainly void and hence can be deleted together with its rank and name. Restore the actual heights h_x and h_y of x and y by means of a search for the largest valid items in $rank(x)$ and $rank(y)$. Link by rank x and y, associating to the new link the ordinal number $i_{max} + 1$. Insert the new rank and the name Z of the resulting set either in $rank(x)$ and $name(x)$ or in $rank(y)$ and $name(y)$, according to the link introduced. Perform an $insert((x, y), w, Q)$ into the Backtracking Queue. As a consequence, at most lgn nodes are deleted from the garbage forest of the Backtracking Queue. Finally, set i_{max} to $i_{max} + 1$

find(x) - starting from node x, follow the most recent valid link leaving the node. The validity of a link can be tested in costant time as described above. Repeat until a node $r(x)$ with no valid outgoing *link* is entered. This node is the root of the tree corresponding to the set containing x. A search for the largest valid item in $name(r(x))$ returns the actual name of the set containing x.

increase_weight(i, α) - perform an $increase(i, \alpha, Q)$ on the Backtracking Queue.

decrease_weight(i, α) - perform a $decrease(i, \alpha, Q)$ on the Backtracking Queue.

$backtrack_1$ - perform a $back_1(Q)$ on the Backtracking Queue.

$backtrack_2(i)$ - perform a $back_2(i, Q)$ on the Backtracking Queue.

The following theorem holds:

THEOREM 3.1: It is possible to perform each *union*, *find*, *increase_weight*, *decrease_weight* and $backtrack_1$ and $backtrack_2$ in $O(lgn)$ time. The space required is $O(n)$.

PROOF: The proof will be given in the full paper.•

4. A LOWER BOUND FOR SET UNION WITH WEIGHTED BACKTRACKING

The bound in Theorem 3.1 is the best possible for a rather large class of algorithms previously defined in the literature [**17, 18, 27**] and known as *nonseparable pointer algorithms*. The computational model is the pointer machine [**22**], and the class does not require the separability assumption (as defined for instance in [**24**]). The only requirement is that the number of pointers leaving each node must be bounded by some constant $c > 0$. As Mannila and Ukkonen pointed out in [**17**], this is not a significant restriction for algorithms which run on a pointer machine (i.e., *pointer based algorithms*). The rules one algorithm must obey to be in such a class are the following:

(i) The operations must be performed on-line.

(ii) Each set element is a node of the data structure. There can be also additional nodes.

(iii) There exists a constant $c > 0$ such that there are at most c edges leaving a node.

(iv) For performing find(x), the algorithm obtains the node v containing x and follows paths starting from v until it reaches the node which contains the name of the corresponding set.

(v) During any union, find, increase_weight, decrease_weight or backtrack operation the algorithm may insert or delete any number of edges. The only restriction is that rule (iii) must hold after each operation.

Our algorithm clearly obeys rules (i), (ii), (iii), (iv) and (v). For the such a class of algorithms, the following lower bound holds.

THEOREM 4.1: Let A be any algorithm which obeys rules (i)-(v). Then there exists a sequence of weighted *unions*, *finds*, *increase_weight*, *decrease_weight*, $backtrack_1$, $backtrack_2$, such that the worst-case per operation time complexity of A is $\Omega(lgn)$.

PROOF : By reduction to the partially persistent set union problem [**17**]. Details will be given in the full paper.•

5. AMORTIZED ANALYSIS

In this section, we will refine the algorithms presented in section 3 in order to get better amortized bounds.

Such solution relies on the use of an extension of *Fibonacci Heaps* [5], which we call *Backtracking Fibonacci Heaps* (*BF-heaps* for short). A BF-heap maintains a collection of items of a set $S = \{s_1, s_2, \ldots, s_n\}$, where each item s has an associated value $v(s)$, under a sequence of the following operations.

$insert(s, v)$: insert item s with $v(s) = v$ in the heap: assign to s an integer $p(s)$, referred to as its position in the queue.

$findmax$: return (a pointer to) the element with maximal value in the heap.

$decrease(s, \alpha)$: decrease by α the value $v(s)$.

$increase(s, \alpha)$: increase by α the value $v(s)$.

$back_1$: delete all elements inserted in the heap after the insertion of the maximum.

$back_2(i)$: delete all elements inserted in the heap after the insertion of the element in position i in the queue.

$meld(h_1, h_2)$: meld heaps h_1 and h_2.

We assume that the items belong to a predefined universe of fixed cardinality.

A BF-heap is essentially composed by a (forest of) F-heaps, where each item s stores its associated position $p(s)$. Furthermore, an array ACCESS$[1, \ldots, n]$ is introduced which, for each entry $i \leq i_{max}$, stores a pointer to the item s with $p(s) = i$ (figure 5). This array substitutes the balanced tree introduced in section 3. In general, the structure may store a set of *void* elements, i.e. elements deleted by $Back_1$ and $Back_2$ operations but still not removed from the heap. A void element s can be identified since $p(s) > i_{max}$. While i_{max} represents the number of non void elements in the structure, let I_{MAX} be the *total* number of elements stored.

In order to obtain better amortized bounds, we substitute BQ's with BF-heaps, in the sense that all operations performed on BQ's are now performed on BF-heaps. It is possible to state the following lemma.

LEMMA 5.1: If we begin with no $BF-heaps$ and perform an arbitrary sequence of k operations, with $n \leq k$ *insert* and $m \leq k$ *decrease*, the total time complexity of such a sequence is $O(k + (m + n)\ lgn)$. Hence, it derives that *findmax*, $back_1$, $back_2$, *increase* and *meld* present an $O(1)$ amortized time complexity, while *decrease* and *insert* present an $O(lgn)$ amortized time complexity.

PROOF: The proof will be given in the full paper.●

The following theorem easily derives from Lemma 5.1 and from Theorem 3.1.

THEOREM 5.1: It is possible to perform each $backtrack_1$, $backtrack_2$ and *increase_weight* in $O(1)$ amortized time and *find*, *decrease_weight* and *union* in $O(lgn)$ amortized time. The space required is $O(n)$.

A somewhat better amortized bound for *find* as the ratio of finds to unions and backtracks increases can be also obtained as suggested by Tarjan (personal communication, 1988), by using the data structure described in [**27**] in combination with Fibonacci heaps and following the same approach described above. This results in a $O(lgn/max\{1, lg(\gamma lgn)\})$ amortized bound for finds, where γ is the ratio of the number of finds to the number of unions and backtracks in the sequence.

6. CONCLUSION

In this paper, we considered an extension of the set union problem, in which dynamic weighted backtracking over sequences of unions is permitted. In particular, we studied the problem of dealing efficiently with weighted unions, subject to dynamic updates (both increases and decreases), when two different kind of backtracking are allowed.

We presented a new data structure which can support each operation in $O(lgn)$ time in the worst case. We proved that this bound is tight for a rather general class of algorithms (known as nonseparable pointer algorithms). Furthermore, we showed also how a different data structure can achieve better amortized bounds. The space complexity of both our data structures is $O(n)$.

ACKNOWLEDGMENTS

We are indebted to Giorgio Ausiello for his encouragement and for many useful suggestions and comments. The second author is also very grateful to Zvi Galil and Raffaele Giancarlo for helpful conversations.

REFERENCES

[**1**] A.V.Aho, J.E.Hopcroft, J.D.Ullman: "*The design and analysis of computer algorithms*", Addison-Wesley, (1972)

[**2**] G.M.Adelson-Velskii, Y.M.Landis: "*An algorithm for the organization of the information*", Soviet. Math. Dokl. **3**, 1259–1262, (1962)

[**3**] N.Blum: "*On the single operation worst-case time complexity of the disjoint set union problem*", SIAM J. Comput. **15**, 1021–1024, (1986)

[**4**] B.Bollobas, I.Simon: "*On the expected behaviour of disjoint set union problems*", 17th ACM Symp. on Theory of Computing, 224–231, (1985)

[**5**] M.L.Fredman, R.E.Tarjan: "*Fibonacci Heaps and their uses in Improved Network Optimization Algorithms*", J. ACM **34**, 596–615, (1987)

[**6**] C.Gaibisso, G.Gambosi, M.Talamo: "*A Partially Persistent Data Structure for the Set-Union problem*" IASI Tech. Report **R.201**, (1987)

[**7**] G.Gambosi, G.F.Italiano, M.Talamo: "*Getting back to the past in the Union-find problem*", Proc. 5th STACS, 8–17, (1988)

[**8**] G.Gambosi, G.F.Italiano, M.Talamo: "*Worst case analysis of the set union problem with extended backtracking*", Theoretical Computer Science, to appear (1988)

[**9**] H.N.Gabow, R.E.Tarjan: "*A linear time algorithm for a special case of disjoint set union*", 15th ACM Symp. on Theory of Computing, 246–251, (1983)

[10] C.J.Hogger: "*Introduction to logic programming*", Academic Press,(1984)

[11] J.E.Hopcroft, J.D.Ullman: "*Set merging algorithms*", SIAM J. Comput. **2**, 294–303, (1973)

[12] T.Ibaraki: "*M-depth search in branch and bound algorithms*", Int. J. Comp. Inform. Sc., **7**, **4**, 313–373, (1978)

[13] M.Loebl, J.Nešetřil: "*Linearity and Unprovability of set union problem*", 20th ACM Symp. on Theory of Computing, 360–366, (1988)

[14] H.Mannila, E.Ukkonen: "*The set union problem with backtracking*", Proc. 13th ICALP, 236–243, (1986)

[15] H.Mannila, E.Ukkonen: "*On the complexity of unification sequences*", 3rd Conf. on Logic Programming, 122–133, (1986)

[16] H.Mannila, E.Ukkonen: "*Space-time optimal algorithms for the set union problem with backtracking*, Tech. Rep. C-1987-80, Departement of Computer Science, University of Helsinky, Finland, (1987)

[17] H.Mannila, E.Ukkonen: "*Time parameter and arbitrary deunions in the set union problem*", Tech. Rep. A-1988-4, Department of Computer Science, University of Helsinky, Finland, (1988)

[18] K.Mehlhorn, S.Naher, H.Alt: "*A Lower Bound for the Complexity of the Union-Split-Find Problem*", 14th ICALP, 479–488, (1987)

[19] J.Nievergelt, E.M.Reingold: "*Binary search trees of bounded balance*", SIAM J. Comput. **2**, 33–43, (1973)

[20] J.Pearl: "*Heuristics*", Addison-Wesley, Reading, MA, (1984)

[21] R.E.Tarjan: "*Efficiency of a good but not linear set union algorithms*", J.ACM **22**, 215–225,(1975)

[22] R.E.Tarjan: "*A class of algorithms which require non linear time to maintain disjoint sets*", J.Computer and System Sciences **18**, 110–127, (1979)

[23] R.E.Tarjan: "*Amortized computational complexity*", SIAM J. Alg. Discr. Meth. **6**, 306–318, (1985)

[24] R.E.Tarjan, J. van Leeuwen: "*Worst-case analysis of set union algorithms*", J. ACM **31**, 245–281, (1984)

[25] J. van Leeuwen, T. van der Weide: "*Alternative path compression techniques*", Tech.Rep. RUU-CS-77-3, Rijksuniversiteit Utrecht, The Netherlands, (1977)

[26] D.H.D.Warren, L.M.Pereira: "*Prolog – the language and its implementation compared with LISP*", ACM SIGPLAN Notices **12**, 109–115, (1977)

[27] J.Westbrook, R.E.Tarjan: "*Amortized Analysis of algorithms for Set Union with Backtracking*", Tech.Rep. TR-103-87, Department of Computer Science, Princeton University, (1987)

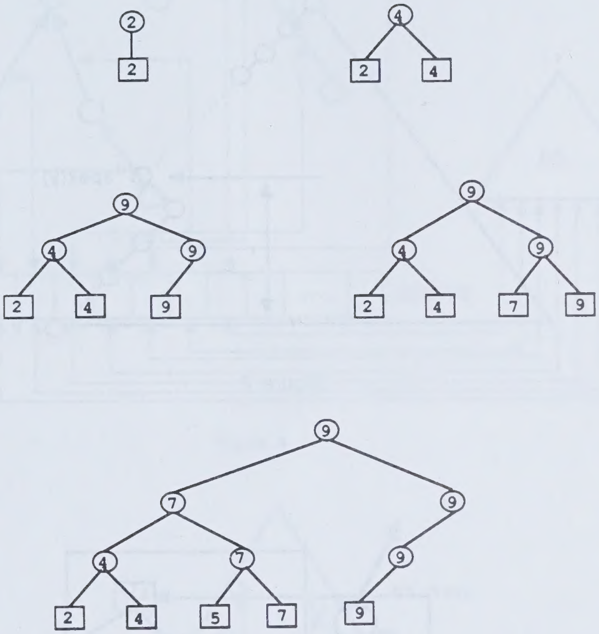

2
2
4
2
4
9
4
9
2
4
9
9
4
9
2
4
7
9
9
7
9
4
7
9
2
4
5
7
9

Figure 1

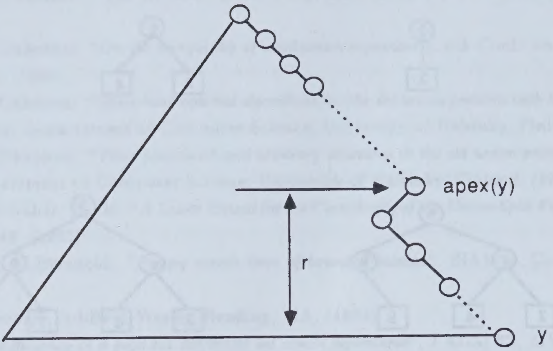


Figure 2

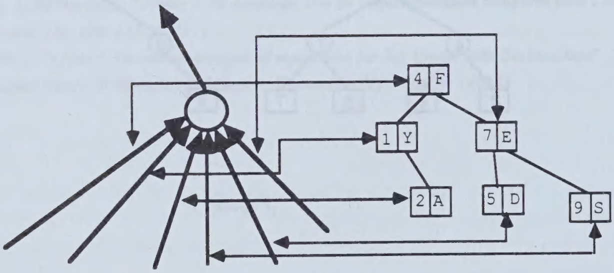

4 F
1 Y
7 E
2 A
5 D
9 S

Figure 3

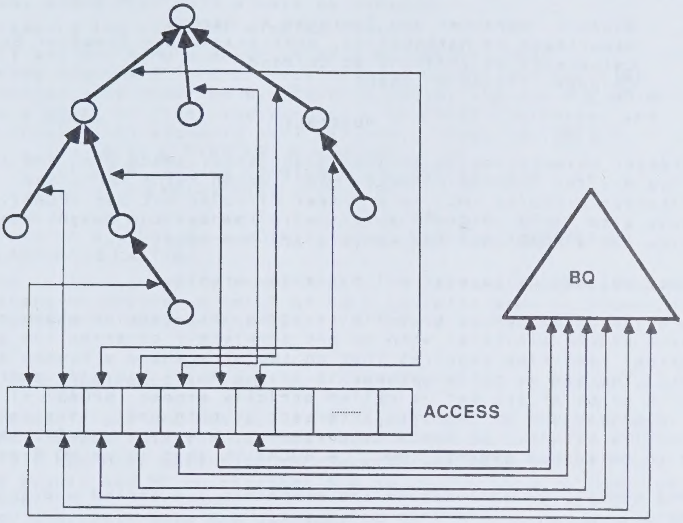
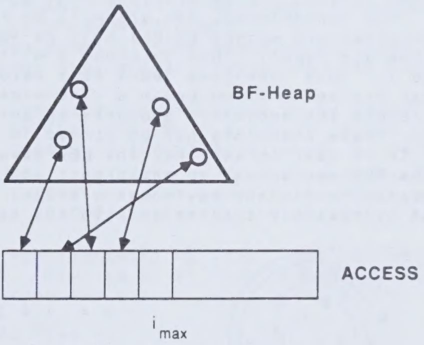


Figure 4



BF-Heap
ACCESS
i_{max}

Figure 5

FINDING NORMAL FORMS, CLIQUES, HAMILTONIAN PATHS, AND HAMILTONIAN CIRCUITS IN A SET OF PROPER CIRCULAR ARCS IN OPTIMUM TIME AND SPACE

Glenn K. Manacher and Terrance A. Mankus
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
Chicago, Illinois 60608

Abstract

We present asymptotically optimal algorithms, space O(n) and time O(n log n), for finding cliques, Hamiltonian Paths (HP), and Hamiltonian Circuits (HC) in a Proper Circular Arc Set (**PCAS**). Our methods also yield an $O(n^2)$ solution to the maximum weight clique problem for a **PCAS** when the weights are non-negative.

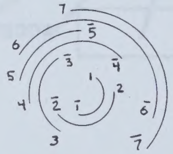
Brevity obliges us to omit all but a few proofs.

A circular arc graph is by definition the intersection graph of a set of arcs around a circle, with no arc completely covering the circle. A further condition requires that no two arcs share a common end point. The set is called **proper** if no arc fully contains another arc. A circular arc set is called **strictly proper** (SPCAS) if the set is proper and no two arcs intersect at both ends, thus covering the entire circle. If two arcs do cover the entire circle, they are said to **horseshoe overlap** (HO). A SPCAS is said to be **HO Free.**

In the sequel, we will assume the endpoints are sorted and that none coincide.

If we sweep around the circle in a clockwise direction, we will encounter arc starting points and arc ending points. If the set has n arcs, we will label or index the arc starting points by unbarred symbols, from 1 to n. These starting points will be known as **left ends** (LE). For convenience, the arc will be referred to by it's LE label. The other arc ending points will be labelled by the barred symbol of the arc label. Thus interval i will have LE labelled i and RE labelled i. This labelling leads to a natural representation of the circular arc set. If we begin a clockwise scan at LE of interval 1, we can record the endpoints encountered until we return to LE of interval 1. These endpoints may be stored in an array of length n, called S. It is easy to see that the LEs appear in order from 1 to n, and that the REs may appear in any order. If the set is **proper**, the REs also appear in increasing, numeric order, beginning with the RE of interval 1, possibly cycling back to the beginning of the array.

Example:



$S = 1\ \bar{4}\ 2\ \bar{6}\ \bar{7}\ \bar{1}\ 3\ \bar{2}\ 4\ 5\ 6\ \bar{3}\ \bar{5}\ 7$

A theorem due to Golumbic [G 80] states that "If G is a proper circular arc graph, then G has a proper circular arc representation in which no two arcs share a common endpoint and no two arcs together cover the entire circle." This representation may be obtained by a naive implementation that takes time $O(n^3)$ and space $O(n)$. However, given our (pre-indexed) input array S, a time $O(n)$ algorithm exists.

From now on, space $O(n)$ will always be assumed.

We first require two key, yet simple, lemmas.

LEMMA i If any arc A HO's a set of arcs B, $|B| \geq 0$, then scanning rightward in S cyclically from LE(A), the next $|B|$ REs encountered comprise B. In this sense, the set $B \equiv HO(A)$ forms a <u>block</u> of right end points, with FIRST (leftmost) and LAST (rightmost) elements well defined. Thus, for $|B| \geq 1$,

(1) $LE(A) < FIRST(B) \leq LAST(B)$,
(2) $b \in B$ implies $FIRST(B) \leq b \leq LAST(B)$, and
(3) no $c \notin HO(A)$ lies within B.

LEMMA ii Let A and A' be arcs in S, with A' thought of as to the left of A. Let $B' = HO(A')$. Then $FIRST(B') \leq FIRST(B)$ and $LAST(B') \leq LAST(B)$.

We note the following easily proven fact: A necessary and sufficient test to determine whether a set S of $n \geq 2$ circular arcs is proper is simply to determine whether the REs in S form a cyclic permutation of $\langle 1,2,..,n \rangle$.

Algorithm SP -- Converts a Proper Arc Set to a Strictly Proper Arc Set.

```
 1)  Determine if any arc HOs arc 1. If none, GOTO 1c.  (LE symbol
     of arc 1 appears in the leftmost cell of array S).
 1a) If arc 1 has HOs, determine LE_block(1) and RE_block(1), sets
     of LE points and RE points that are in cyclic numeric order as
     follows:
     For RE_block(1): find the leftmost RE in S.  Call it LM_RE.
     Find the rightmost RE in S that arc 1 HOs.  Call it RM_RE.
     Store in RE_block(1) the RE symbols for LM_RE to RM_RE, in
     cyclic numeric order.
     For LE_block(1): find the first LE point to the right of
     RM_RE in S.  Call it FR_LE.  Store in LE_block(1) the LE
     symbols from 1 to FR_LE in numeric order.
 1b) Reform S by writing the symbols in RE_block(1) followed by
     LE_block(1) in the order they appear, beginning with cell 1 of S.
     Arc 1 is now HO Free.
 1c) For conceptual ease, create a working array S' by rewriting S into
     S', in cyclic order, so that the LE of arc 1 is in the rightmost
     cell of S'.  The LEs appear in the order 2,..., n, 1, in S'.
 2)  Using LEMMA i, begin the final scan of S'.  Look at successive
     LEs in S' from right to left, beginning with the LE of arc n,
     checking for HOs.  Let LE(CUR) represent the currently scanned
     LE symbol.
 2a) If $HO(CUR) \neq \phi$ find the rightmost RE in S' that LE(CUR)'s
     HOs.  Call this RM_RE.  Rewrite S' so that LE(CUR) is removed
     from its current position and placed immediately to the right of
     RM_RE.
 2b) If CUR was arc 2, GOTO Step 3.
 2c) Reset LE(CUR) as the next LE in the right to left scan of S',
     and repeat Step 2.
 3)  S' is now totally HO Free.  Cyclically rewrite S' back into S so
     that LE of arc 1 is in the leftmost cell of S.
```

Example: $S = 1\ \overline{4}\ 2\ \overline{5}\ 3\ \overline{6}\ 4\ 5\ 6\ \overline{1}\ \overline{2}\ \overline{3}$ LE_block(1) = $1\ 2\ 3$

Reformed: $S = \overline{4}\ \overline{5}\ \overline{6}\ 1\ 2\ 3\ 4\ 5\ 6\ \overline{1}\ \overline{2}\ \overline{3}$ RE_block(1) = $\overline{4}\ \overline{5}\ \overline{6}$

Final Form: $S' = 2\ 3\ 4\ 5\ 6\ \overline{1}\ \overline{2}\ \overline{3}\ \overline{4}\ \overline{5}\ \overline{6}\ 1$

At this stage, the input array is now **HO** Free and it represents a **SPCAS**.

THEOREM 0 Algorithm SP accepts a Proper Arc Set and produces a Strictly Proper Arc Set in time O(n) and space O(n).

We need a few lemmas for the main proof.

LEMMA A The output of algorithm SP is a proper set.

LEMMA B Algorithm SP preserves all interval intersections.

LEMMA C Algorithm SP creates no new intersections.

LEMMA D The output of algorithm SP is HO Free.

LEMMA E Algorithm SP use space O(n).

LEMMA F Algorithm SP takes time O(n) to complete.

All these lemmas can be proven without difficulty from [G 80] and **LEMMA i** and **LEMMA ii**.

Proof of Theorem 0: Lemma A, Lemma B, Lemma C, Lemma D, Lemma E, and Lemma F.

We are now in a position to present two of the main results of this paper. (Note: both these results depend upon the array S representation of the arc set to achieve the desired time complexity.)

HP for PCAS Theorem A Hamiltonian Path can be found, or proved nonexistent, in time O(n) for a Proper Circular Arc Set.

Proof: Use algorithm SP to obtain a **SPCAS**. Scan the resulting **SPCAS** for gaps. (Finding gaps in a **CAS** is essentially trivial; the reader can easily find a time O(n) algorithm.) If there are no gaps, connect intervals in numeric order for the **HP**. If there is one gap, begin with first arc clockwise past the gap, and connect in cyclic numeric sequence. This is the **HP**. If there are two or more gaps, no **HP** exists.

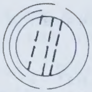
HC for PCAS Theorem A Hamiltonian Circuit can be found, or proved nonexistent, in time O(n log n) for a Proper Circular Arc Set.

Proof: Use algorithm SP to obtain a **SPCAS**. Scan the resulting **SPCAS** for gaps. If there are no gaps, connect intervals in numeric order for the **HC**. If there is one gap, the **SPCAS** forms a set of intervals, so use the main result of Manacher and Mankus[MM 87] (See also Bertossi [B 83]) to determine the **HC** or nonexistence of the **HC**. If two or more gaps are found, no **HC** is possible.

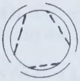
CLIQUES

It is easy to see that a strictly proper circular arc graph can be converted to a circle chord graph (**CCG**) by connecting the end points of the arcs, thus creating chords. The intersection of two arcs in the proper arc set causes the corresponding chords to "cross" (intersect) inside the circle. If the initial arc set is strictly proper, then pairwise nonintersecting arcs have corresponding pairwise parallel chords. It is crucial to note that a **CCG** from a **SPCAS** must have the property that no three chords are mutually parallel (For three or more chords to be mutually parallel, they must be non-intersecting and it must be possible to intersect all of them with a fourth chord). This is true because 3-parallelism of chords forces containment of at least one of the corresponding arcs in another.

Example:



Arcs and 3 parallel chords



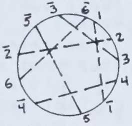
Arcs and 3 non-parallel chords

Algorithms for the clique problem for chords or circle graphs have been presented by Rotem and Urrutia [RU 81], and also by Buckingham [BU 81] who gave an $O(n^2)$ algorithm for the unweighted case. Quite recently, Apostolico, Atallah, and Hambrusch [AAH 87] have given an $O(n \log n + \min(e, (nq)\log(2n/q)))$ algorithm where e is the number of edges and q is the cardinality of the largest clique. These are the best known results. The algorithm presented here will find an unweighted chord clique in time $O(n(L+\log n))$ where L is the length of the longest increasing subsequence among right endpoints when the chords are represented as intervals. This algorithm is often dominated by [AAH 87], since $q \leq L$. However, for an "average" CCG model, in which the endpoints are placed on the circle perimeter at random, it may be shown that $q \cong L/2$ and $L = c*n^{1/2}$, where $2 \leq c \leq 3.84$ [SS 84], so that [AAH 87] requires time $O[n^{3/2} \log n]$ while our algorithm takes time $O[n^{3/2}]$.

We will present, in the following lemmas, theorems, and algorithms, a new approach to the clique problem that has no better time complexity than $O(n(L+\log n))$, but which is adaptable to the special circle graph models that result from strictly proper circular arcs, in which case we will obtain an $O(n \log n)$ solution to the clique problem.

Using our modified input array S (now representing a **SPCAS**), we create our corresponding circle chord model. We will store the resulting model in an array of length 2n, denoted by σ. The remodeling (i.e. re-indexing) is accomplished by a trivial left to right, one time pass of the array S. We start with **LE** of interval 1 in S. The first occurrence of an arc's endpoint, either left or right, is re-indexed as the **LE** of the next available chord index, and the other endpoint is re-indexed as the chords **RE**.

Example:



$$S = 1 \quad \bar{5} \quad \bar{6} \quad 2 \quad \bar{1} \quad 3 \quad \bar{2} \quad 4 \quad 5 \quad \bar{3} \quad 6 \quad \bar{4}$$

$$\sigma = 1 \quad 2 \quad 3 \quad 4 \quad \bar{1} \quad 5 \quad \bar{4} \quad 6 \quad \bar{2} \quad \bar{5} \quad \bar{3} \quad \bar{6}$$

Note that σ is a sequence of unbarred (LE) and barred (RE) symbols. It has become an interval model. The unbarred symbols appear in numeric order from left to right. The barred symbols can appear in any order, except that every barred symbol must appear to the right of the corresponding unbarred symbol. We will denote the sequence of barred symbols by $\bar{\sigma}$, and in the sequel, we will use $\bar{\sigma}$ as the basis for our indexing.

It is very important to note that, when dealing with the interval model derived from a **PCAS**, that $\bar{\sigma}$ has no left to right decreasing subsequences of length 3 or more. This is due to the no-3-parallel-chord property already noted.

Let L be the longest increasing subsequence (LIS) in $\bar{\sigma}$.

For $1 \leq i \leq n$, and $1 \leq k \leq L$, define $U_{i,k}$ to be a decreasing sequence (as k increases for a fixed i) of positions in $\bar{\sigma}$ representing the location of the rightmost left endpoint in $\bar{\sigma}$ of an increasing subsequence of length k considering only the elements {1,2,..,i}.

Pos(i) is the position of symbol i in $\bar{\sigma}$. Pos*(i) is the position in $\bar{\sigma}$ of the nearest barred symbol in σ to the right of (unbarred) i in σ.

$L(i)$ is the length of the longest increasing subsequence of $\overline{\sigma}$ formed entirely from elements of the set $\{1,2,..,i\}$.

Let K_s, $1 \leq s \leq L$, be integers, called keys, such that
a) $K_1 = 1$,
b) $K_q < K_r$ when $q < r$, and
c) $L(K_s) = L(K_s - 1) + 1$.

Let $K = \{K_1, K_2, ..., K_L\}$. In words, K is the unique ascending subsequence of $\{1,2,..,n\}$ in which $i = K_s$ implies that there exists an ascending subsequence of $\overline{\sigma}$ of length s consisting entirely of symbols in $\{1,2,..,i\}$, whereas there exists no ascending subsequence of length s of $\overline{\sigma}$ consisting entirely of symbols in $\{1,2,..,i-1\}$.

Hunt and Szymanski [HS 77] found K by constructing $T_{i,k}$ for all i, where $T_{i,k}$ is an increasing sequence of positions in $\overline{\sigma}$ such that $T_{i,k}$ is the leftmost right endpoint of any subsequence of length k in $\overline{\sigma}$ restricted to the elements $\{1,2,..,i\}$. If no such subsequence exists, then $T_{i,k}$ is "blank".

Let $T_{i,*}$ denote the ascending sequence $T_{i,1}, T_{i,2}, ..., T_{i,L(i)}$, where $L(i)$ is clearly the largest number such that $T_{i,k}$ is nonblank.

LEMMA 1 [F 75][HS 77] The set of keys $K = \{K_1, K_2, ..., K_L\}$ can be found in time $O(n \log n)$.

This is done by creating a left to right increasing stack sequence, obtained during a left to right scan of $\overline{\sigma}$. Our stacks grow downward, so the last placed element is at the bottom of a stack column. When completed, the keys are found at the bottoms of the stack columns.

Example $\overline{\sigma} = \overline{1}\ \overline{4}\ \overline{2}\ \overline{5}\ \overline{3}\ \overline{6}$

```
1     1 4    1 4    1 4 5    1 4 5    1 4 5 6
               |      |        | |      | |
               2      2        2 3      2 3
```

The keys are 1,2,3, and 6

FACT 1 $L(i) = s$, where K_s is the largest key for which $K_s \leq i$.

Definition $U_{i,*}$ denotes the descending sequence $U_{i,1}, U_{i,2}, .., U_{i,L(i)}$.

FACT 2 The keys for $U_{i,k}$ are the same as for $T_{i,k}$.

LEMMA 2 Let $m \in U_{i,*}$. Then $m \in U_{j,*}$ for all $j \geq i$.
Note: This is a crucial lemma, and has a rather subtle combinatorial proof. The corresponding property for $T_{i,*}$ does not hold.

LEMMA 3 $U_{i,*}$ has the following form:

a) $U_{1,*} = pos(1)$,

b) $U_{i,*} = U_{i-1,*}$ if i is not a key, and

c) $U_{i,*}$ is derived from $U_{i-1,*}$ by the interpolation of one new element, called $ne(i)$, if i is a key.

Definition Let i be a key, say K_s. Let $ne(K_s)$ be the new element in $U_{K_s,*}$, not in $U_{K_s-1,*}$. Call P_s the **pivot** element where $ne(K_s) \equiv pos(P_s)$. Then we will say that P_s is **paired** with K_s.

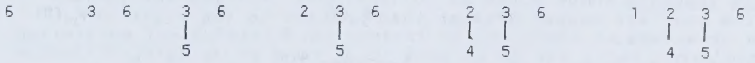
FACT 3 $U_{K_s,*}$ consists of $pos(P_1)$, $pos(P_2), \ldots, pos(P_s)$ in decreasing numeric order.

LEMMA 4 $U_{i,k}$ is completely characterizable by the key-pivot pairings (K_s, P_s), $1 \le s \le L$.

LEMMA 5 The set $\{P_1, P_2, \ldots, P_L\}$ can be found in $O(n \log n)$.

Method: Simply scan from right to left, constructing the right to left decreasing stack sequence, again with the last placed as the bottommost element in a stack column. The pivots are first elements in the stacks, seen as the stack column tops in this structure.

Example $\bar{\sigma} = \bar{1}\ \ \bar{4}\ \ \bar{2}\ \ \bar{5}\ \ \bar{3}\ \ \bar{6}$



The pivots are 1, 2, 3, and 6.

CRUCIAL NOTE Knowing K and $P=\{P_1, P_2, \ldots, P_L\}$ does not determine the key-pivot (K-P) pairings. Finding the pairings in time less than $O(nL)$ is a major unsolved problem; an $O(nL)$ algorithm is known to us using the methods of [BU 81].

We now motivate the study of $U_{i,k}$. It is well known (e.g. [G 80], and [BU 81]) that determining a maximum clique in a circle graph model (a largest clique under intersection of a set of circle chords) is equivalent to finding a maximum clique under overlap (intersection without containment) in σ , which is equivalent to find a pair of increasing sequences u, $\bar{u}$ in σ such that

a) u is a string of unbarred symbols, e.g. 1 3 6,

b) $\bar{u}$ is identical to u except that it is composed entirely of barred symbols, e.g. $\bar{1}\ \bar{3}\ \bar{6}$, and

c) u and $\bar{u}$ must span disjoint regions of σ.

LEMMA 6 If there exist a subsequence $u\,\bar{u}$ in σ such that $|u| = h$, then there exists a number j for which $U_{j,h} \ge pos*(j)$. The converse is also true: $U_{j,h} \ge pos*(j)$ implies the existence of a subsequence $u\,\bar{u}$ in σ , where $|u| = h$.

THEOREM 1 If the maximum overlap clique of intervals has size m, then

$$m = \max_j \{h \mid U_{j,h} \ge pos*(j)\}.$$

THEOREM 2 If, given $\bar{\sigma}$, the K-P pairing can be found in space $O(PS(n))$ and time $O(PT(n))$, then the maximum overlap clique can be found in space $O(n+PS(n))$ and time $O(n \log n + PT(n))$.

LEMMA 7 Theorem 1 can be improved to $m = \max_{j \in K}\{h \mid U_{j,h} \ge pos*(j)\}$.

Note: Lemma 7 does not improve the asymptotic complexity of our results.

LEMMA 8 The equivalent chord model (ECM) of a set of strictly proper arcs has intersecting chords only if the arcs intersect.

LEMMA 9 The ECM cannot have three parallel chords.

LEMMA 10 No 3 parallel chords inplies maximum pile depth 2 in the interval model. (Call this property NPD3 - "No Pile Depth 3 or more". A pile is a set of intervals in which every interval contains every other interval smaller than itself.)

LEMMA 11 NPD3 implies that $\bar{\sigma}$ has maximum left to right decreasing subsequence of at most length 2, thus no pile of depth > 2.

LEMMA 12 Any interval model σ with the NPD3 property has a right endpoint subsequence $\bar{\sigma}$ that consists of an interleaving of two disjoint increasing subsequences. Conversely, any interleaving of two disjoint increasing sequences A and B, such that $i \in \{1,2,..,n\}$ appears exactly once in either A or B, is an example of $\bar{\sigma}$ with the NPD3 property.

Definition Given a permutation π of $\{1,2,..,n\}$. Let the primary increasing sequence $\Psi(\pi)$ be given as follows: $\Psi_1(\pi) = 1$ and for $j > 1$, $\Psi_j(\pi)$ is the smallest number greater than $\Psi_{j-1}(\pi)$ and to the right of $\Psi_{j-1}(\pi)$ in π, if it exists.

Examples: If π = 3 6 1 4 2 7 5, then $\Psi(\pi)$ = 1 2 5
If π = 6 1 3 2 7 4 5, then $\Psi(\pi)$ = 1 2 4 5

Definition Given a permutation π, let the secondary sequence $\varphi(\pi)$ consist of all the symbols of π in order, but with the symbols in the primary sequence, $\Psi(\pi)$, deleted.

Example: if π = 3 6 1 4 2 7 5 then $\varphi(\pi)$ = 3 6 4 7.

LEMMA 13 If $\bar{\sigma}$ has the NPD3 property, then $\varphi(\bar{\sigma})$ is an increasing sequence.

Definition Let Rn(π,i) (Rn means "restriction") be the subsequence of π consisting of the elements of $\{1,2,..,i\}$ written in the same order as they occur in π.

LEMMA 14a If e is in the primary subsequence of $\bar{\sigma}$, then it lies rightmost in Rn($\bar{\sigma}$,e).

LEMMA 14b Every e in the primary sequence is a key of $\bar{\sigma}$.

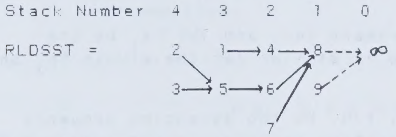
LEMMA 14c Every e in the primary sequence is also a pivot. In fact, every such e is a key-pivot self-pair e:e in the K-P pairing.

Notation The right to left stack column construction used previously to discover the pivots will be called a Right to Left Decreasing Schensted Stack Sequence, **RLDSSS** (Named after C. Schensted, who we believe first devised this structure). The stack columns are assigned the numbers 1 to L, from right to left.

LEMMA 15 [F 75] Let $e \in \pi$ and let e appear on stack number m in the RDLSSS for π. Then the length of the longest right to left decreasing subsequence in π ending in e is m. This is also the length of the longest increasing left to right subsequence in π beginning with e. We denote this quantity L(π,e).

Definition A Right to Left Decreasing Schensted Stack Tree, RLDSST, is a RLDSSS with one more piece of information per element: For each element e in the stack m, m > 1, there is a back-link to the element f(e) given by the rule : f(e) is the smallest number in stack m-1 larger than e. The elements in stack 1 are all back-linked to an imaginary element ∞ in "stack zero".

Example: If $\overline{\sigma}$ = 3 9 2 5 7 1 6 4 8 then RLDSST($\overline{\sigma}$) is as as shown below.



Corollary 1 The elements of $U_{K_s,*}$ are the topmost elements of RLDSST(Rn($\overline{\sigma}$,K_s)).

Definition The RLPath(π,e) consist of the right to left path in the RLDSST(π) from ∞ to e in π. Root(e) is the first (rightmost) element of RLPath(π,e). The root is of course in the first stack of RLDSST(π).

LEMMA 16 The sequence of elements in RLPath(π,e) is the lexically rightmost right to left decreasing subsequence of length L(π,e) ending in e.

Note: By lexically rightmost we mean that the rightmost (first) element of RLPath(π,e) is as far right as possible. Fixing that element, the second rightmost element is as far right as possible, and so on, with each successive element being rightmost of all the elements remaining.

FACT 4 In RLDSST(Rn($\overline{\sigma}$,i)), i is the bottommost element of stack 1.

Definition For any permutation π, let $d_\pi(e,i) \equiv L(Rn(\pi,i),e)$.

FACT 5 $d_\pi(e,i+1) = d_\pi(e,i)$ or $d_\pi(e,i) + 1$.

LEMMA 17 In RLDSST(Rn($\overline{\sigma}$,K_s)), there is a path from K_s to P_s.

LEMMA 18 Let T_i = RLDSST(Rn(π,i)), i>1. For all $e \in T_i$ such that root(e) = i, $d_\pi(e,i) = d_\pi(e,i-1) + 1$, and for all $e \in T_i$ such that root(e) $\neq$ i, $d_\pi(e,i) = d_\pi(e,i-1)$.

LEMMA 19 The path from K_s to P_s, guaranteed by Lemma 17, cannot contain any P_y for y < s. Note that P_y is the matching pivot for K_y for y < s.

Definition Suppose $\overline{\sigma}$ has the NPD3 property. By Lemma 13, all e are either primary ($e \in \Psi(\overline{\sigma})$) or secondary ($e \in \varphi(\overline{\sigma})$). Define a key or a pivot, as primary if it belongs in the primary sequence of $\overline{\sigma}$, secondary otherwise.

LEMMA 20 All secondary keys are matched with secondary pivots.

LEMMA 21 Let Path(K_s,P_s) be the key-pivot path in RLDSST(Rn($\widetilde{\sigma}$,K_s)) from K_s to P_s guaranteed by Lemma 17. Suppose that K_s is secondary. Then:

a) P_s is secondary,

b) Path(K_s,P_s) contains no element in $\Psi(\bar{\sigma})$ (no primary element), and

c) Path(K_s,P_s) is a strictly descending sequence in $\varphi(\bar{\sigma})$; that is, it contains exactly (and only) the elements of $\varphi(\bar{\sigma})$ lying between K_s and P_s inclusive.

Notation Let Kp_i be the ith primary key, and let Ks_j be the jth secondary key. The meaning is similar for the pivots Pp_i and Ps_j.

LEMMA 22 Let Ks_1, Ks_2, ..,Ks_t, t>0, be the ascending sequence of secondary keys. Then the matching secondary pivot sequence Ps_1, Ps_2, ..,Ps_t is also an ascending sequence.

THEOREM 4 Let $\widetilde{\sigma}$ have the NPD3 property. Then the key-pivot pairing can be computed in space O(n), time O(n log n), as follows:

1) Isolate the primary sequence Ψ and the secondary sequence φ , using standard techniques with balanced trees.
2) Find the set K in time O(n log n).
3) Find the set P in time O(n log n).
4) For each $e \in \Psi$, place e:e in set PAIRS.
5) Find $Ks \equiv K-\{e \in \Psi\}$ and $Ps \equiv P-\{e \in \Psi\}$.
6) Sort Ks and sort Ps.
7) Pair the smallest element of Ks with the smallest element of Ps, the next smallest Ks with the next smallest Ps, and so on. Place K-P pairs in PAIRS.

PAIRS then contains the key-pivot pairings.

The **MAIN RESULT** follows.

THEOREM 5 The following algorithm will find a clique of unweighted proper circular arcs in space O(n) and time O(n log n), given the model: (all these steps require space O(n)).

1) Sort the endpoints of the arcs.
2) Utilize the algorithm of Theorem SP to transform the proper arc set to a strictly proper arc set.
3) Construct a set of circle chords by connecting the endpoints of the strictly proper circular arcs in time O(n).
4) Open up the chord set into a set of intervals I. Reindex the intervals, labelling then by the left to right sequence of their left endpoints, taking time O(n).
5) Using the algorithm of Theorem 4, find the Key-pivot pairings in time O(n log n).
6) Using Theorem 2, find a maximum clique C under overlap of intervals in I in time O(n log n) using the key-pivot pairing from step 5.
7) Identify the intervals in C with chords in the chord set, and then with strictly proper arcs in the strictly proper model, and finally with arcs in the proper arc set model.
   This is the desired maximum clique.

Proof: Application of the above Lemmas and Theorems.

THEOREM 6 [VEB 77] If the proper arc model is presorted, the maximum clique can be found in space O(n) and time O(n log log n).

Finally, application of lemmas 12 and 13 can be shown fairly easily to yield

Theorem 7 A maximum weight clique for a **PCAS** can be found in space O(n) and time $O(n^2)$ when the weights are non-negative.

REFERENCES

[AAH 87] Apostolico, Alberto, Atallah, Mikhail, and Hambrusch, Susanne. "New Clique and Independent Set Algorithms for Circle Graphs". Department of Computer Science, Purdue University, November, 1987.

[B 83] Bertossi, Alan. "Finding Hamiltonian Circuits in Proper Interval Graphs". Inf. Processing Lett. 17, 1983, 97-101.

[BU 81] Buckingham, Mark A. "Efficient Stable Set and Clique Finding Algorithms for Overlap Graphs". Dept. of Computer Science, New York University, 1981.

[F 75] Fredman, Michael L. "On Computing the Length of Longest Increasing Subsequences". Discreet Math 11, 1975, pp. 29-35.

[G 80] Golumbic, Martin. "Algorithmic Graph Theory and Perfect Graphs". Academic Press, 1980. See in particular Theorem 8.18.

[HS 77] Hunt, J. W., and Szymanski, T. G. "A Fast Algorithm for Computing Longest Increasing Subsequences". Comm. of the ACM 20, 5, May 1977.

[MM 87] Manacher, Glenn K. and Mankus, Terrance A. "An Optimum Θ(n log n) Algorithm for Finding a Canonical Hamiltonian Path and a Canonical Hamiltonian Circuit in a Set of Intervals". Proc. of the 25th Annual Allerton Conf. on Communication, Control, and Computing, 1987, pp 24-33.

[RU 81] Rotem, D., and Urrutia, J. "Finding Maximum Cliques in Circle Graphs". Networks, 1981, pp. 269-278.

[SS 84] Saks, G., and Sturtevant, D. Private communication, 1984.

[VEB 77] Van Emde Boas, P. "Preserving Order in a Forest in less than Logarithmic Time and Linear Space". Inf. Proc. Lett. 6, 1977, pp. 80-82.

LINEAR TIME ALGORITHMS ON CIRCULAR–ARC GRAPHS

WEN–LIAN HSU AND KOUHUI TSAI
Department of Industrial Engineering and Management Sciences
Northwestern University, Evanston, Illnois 60208

ABSTRACT

We give a very simple linear time algorithm which solves the following three problems (unweighted case) on circular–arc graphs: the maximum independent set, the minimum clique cover, and the minimum dominating set problems.

1. INTRODUCTION

In [4], Masuda and Nakajima gave an O(n) time algorithm for the (unweighted) maximum independent set problem on circular–arc graphs (assuming a circular arc model is given and the endpoints have already been sorted in the clockwise order). Later, Hsu [3], Manacher (private communication) independently produced simpler alternate algorithms for this problem. In this paper, we present an extremely simple O(n) algorithm which solves the following three problems (unweighted version) on circular–arc graphs simultaneously: the maximum independent set, the minimum clique cover, and the minimum dominating set problems. Our approach takes advantage of the underlying structure of circular–arc graphs which is amenable to greedy algorithms.

An arc family F is a collection of arcs on the unit circle. A graph G with vertex set V and edge set E is said to be a *circular–arc graph* if there exists an arc family F and a one–to–one correspondence between V and F such that two vertices in V are adjacent in G iff their corresponding arcs overlap. For convenience, we shall consider the family F itself rather than its corresponding graph G. A set of arcs is *independent* if no two of them overlaps with each other. The ***maximum independent set*** **(*MIS*)** ***problem*** on an arc family F is to determine a maximum size (denoted by $\alpha(F)$) independent set in F. A ***clique*** in F is a set of mutually overlapping arcs. The ***minimum clique cover*** (MQC) problem is to determine a minimum size (denoted by $\theta(F)$) collection of cliques whose union is F. A set D in F is a ***dominating set*** if every arc in F\D overlaps with some arc in D. The ***minimum dominating set*** (MDS) problem is to determine a minimum size (denoted by $\tau(F)$) dominating set in F.

Without loss of generality, assume all arc endpoints are distinct and no arc covers the entire circle. Now, label the n arcs arbitrarily from 1 through n. Label the endpoints from 1 to 2n according to their clockwise order. Denote an arc that begins at endpoint p and ends at endpoint q in the clockwise direction by (p,q).

Define p to be the *head* (or counterclockwise endpoint) of the arc and q to be the *tail* (or clockwise endpoint). An example is shown in Figure 1.

The continuous part of the circle which begins with endpoint c and ends with d in the clockwise direction is referred to as *segment* **(c,d)** of the circle. We use "arc" to refer to a member of F and "segment" to refer to a part of the circle between two endpoints. By this definition, an arc (p,q) of F is also a segment (p,q). A point on the circle is said to be in arc (p,q) if it falls within the interior of segment (p,q).

An arc i of F is said to be ***properly contained in*** another arc j if both endpoints of i are contained in arc j. An arc family F is said to be ***proper*** if no arc in F is properly contained in another. Given any arc family F, let C(F) be the collection of arcs which are properly contained in some other arc in F. Define $F' = F \backslash C(F)$ to be the proper subfamily of F. For all three problems, it is easy to show that an optimal solution is contained in a proper subfamily of F. This enables us to define a greedy search algorithm.

We divide our discussion as follows. In the next section, we review O(n) algorithms for the MIS problem. In Section 3, an new O(n) algorithm for the MIS problem is presented. The same algorithm is then used in Section 4 to solve the MQC problem and in Section 5 to solve the MDS problem.

2. AN REVIEW OF O(N) TIME ALGORITHMS FOR THE MIS PROBLEM

The first O(n) algorithm by Masuda and Nakajima involves the following two steps:

(1) Reduce the MIS problem on an arc family F to that of its proper subfamily F'.

(2) Determine a MIS in F' in O(n) time.

Later Hsu [3] simplified their algorithm by removing more arcs in step (1) so that the MIS problem on the resulting subfamily becomes trivial. Below, we briefly describe his approach.

Definition. An arc i is ***contained in*** another arc j if every arc overlapping with j (including j itself) also overlaps with i. An arc family F is *strictly proper* if no arc in F is contained in another.

Clearly, if a MIS contains an arc j, then one can replace j by an arc which is contained in j and obtain another MIS of the same size. Motivated by this, Hsu [] adopted the following approach:

(1) Reduce the MIS problem on an arc family F to that of a strictly proper subfamily F" such that $\alpha(F") = \alpha(F)$ by removing arcs which contain others

one by one.

(2) Determine a MIS of F" in $O(|\alpha(F)|)$ time.

Define a ***head*** (respectively, ***tail***) ***block*** to be a set of maximally contiguous heads (respectively, tail) in the clockwise endpoint order. The term "block" is used to refer to either a head block or a tail block. A strictly proper family F" can be characterized by each of the following two properties:

by each of the following two properties:

(2.1) F" is proper and each block of F" contains exactly one endpoint.

(2.2) Let the arcs in F" be clockwise ordered as $i_1, i_2, \ldots, i_n$. Then there exist an integer k such that arc i_i overlaps with i_j $(i<j)$ iff either $j-i \leq k$ or $n-j+i \leq k$.

Let s be the maximum number of arcs in F" passing through a common point of the circle. Denote |F"| by n. It can be easily shown that $\alpha(F") = \lfloor n/s \rfloor$. Define N(i,F") to be those arcs in F" overlapping with i. Next, we show that $\theta(F") = \lceil n/s \rceil$.

Lemma 2.3. *$\theta(F") = \theta(F)$.*

Proof. We only need to show $\theta(F") \geq \theta(F)$. Suppose the sequence of deletion is $p_1, p_2, \ldots, p_r$, namely, $F" = F\backslash\{p_1,p_2,\ldots,p_r\}$. Now, suppose the lemma holds for $F_1 = F\backslash\{p_1,p_2,\ldots,p_{r-1}\}$. By the deleting rule, there exists an arc $t \in F_1$, such that $N(t,F_1) \subseteq N(p_r,F_1)$. Let C be a MQC for F". Let Q be clique in C containing t. Since $Q \subseteq N(t,F") \subseteq N(t,F_1) \subseteq N(p_r,F_1)$, p_r overlaps with every arcs in Q. Hence, $Q \cup \{p_r\}$ is a clique in F_1 and $(C\backslash\{Q\}) \cup \{Q\cup\{p_r\}\}$ is a clique cover for F_1. Therefore, $\theta(F") \geq \theta(F_1) = \theta(F)$. □

Next, we discuss the size **q** of a maximum clique in a strictly proper family F".

Lemma 2.4. *If $n \leq 2s-1$, then $q = n$; otherwise, $q = s$.*

Proof. Let the arcs of F" be $u_1, u_2, u_3, \ldots, u_n$ arranged according to their clockwise order. The constant k in property (b) is s–1. If $n \leq 2s-1$, then by property (b), $N(u_i,F") = F"\backslash\{u_i\}$ for every i. Hence, F" is itself a clique and q = n.

Now, assume $n > 2s-1$. Obviously, $q \geq s$. We shall show that $q \leq s$. Because F" is highly symmetric, it suffices to show that the maximum clique containing u_s has a size no greater than s. Consider $N(u_s,F") = \{u_1,\ldots,u_{s-1}\} \cup \{u_{s+1},\ldots,u_{2s-1}\}$. Let G_s be the subgraph induced on $N(u_s,F")$. Then the problem is equivalent to show that the size of a maximum independent set in $\overline{G}_s$ is no greater than s–1. This follows from the fact that $\overline{G}_s$ is a bipartite graph and $\{(u_1,u_s),(u_2,u_{s+1}),\ldots, u_{s-1},u_{2s-1})\}$ is a matching of size s–1 in $\overline{G}_s$. □

Lemma 2.5. *If* $n \leq 2s-1$, *then* $\theta(F'') = 1$; *otherwise,* $\theta(F'') = \lceil n/s \rceil$.

Lemma 2.6. *For any arc family F,* $\alpha(F) \leq \theta(F) \leq \alpha(F) + 1$.

Hence, the concept of "strictly proper" arc families can be used to solve the MIS and the MQC problems simultaneously in O(n) time. The best previous algorithm for the MQC problem by Gupta, Lee and Leung [2] takes $O(n\log n)$ time. In the next section, we describe a much simpler algorithm for the MIS problem which can be used to solve the MQC and the MDS problems.

3. THE MAXIMUM INDEPENDENT SET PROBLEM

In this section, we present a simple algorithm for the MIS problem which does not depend on finding a strictly proper family. Below, we describe some properties of proper families which are useful for the algorithm.

Definition. For each arc i, define the following:

NEXT(i) – the arc whose head is the first one encountered in a clockwise traversal from t(i)

GD(i) – the maximal independent set of the form i, i_1, ..., i_k, where i_1 = NEXT(i) and i_t = NEXT(i_{t-1}), t = 2, ..., k (GD is short for GREEDY)

LAST(i) – NEXT(i_k) (note that LAST(i) must overlap with i)

Lemma 3.1 [3]. *Let P be an independent set in a proper family F. Let i be any arc in P. Then* $|P| \leq |GD(i)|$, *namely, GD(i) is maximum among all IS containing i.*

Lemma 3.2 [3]. *Let i be any arc in a proper family F. Let* $P_{\max}$ *be a MIS in F.* Then $|P_{\max}| - 1 \leq |GD(i)| \leq |P_{\max}|$.

Definition. An arc i is called a ***good arc*** if |GD(i)| = α(F); otherwise (by Lemma 3.2, |GD(i)| = α(F)–1), it is called a ***bad arc.***

Consider the directed graph H = (V(H),E(H)) with V(H) = F' and (i,j) $\in$ E(H) iff j = NEXT(i). Each vertex in H has out–degree 1. Define $NEXT^2(i)$ = NEXT(NEXT(i)) and $NEXT^k(i) = NEXT(NEXT^{k-1}(i))$ for k > 2. For any arc i, there must exist indices a < b such that $NEXT^a(i) = NEXT^b(i)$ (consider forming a directed path from i). Hence there must exist at least one directed cycle in H, in particular, there exists a directed cycle consisting of good arcs (if we start with a good arc i). Furthermore, |GD(u)| = |GD(u')| for any two arcs u and u' in a directed cycle.

Theorem 3.3. *Let C_1, C_2 be two disjoint directed cycles in H. Let u, v be two arcs in C_1, C_2, respectively. Then $|GD(u)| = |GD(v)|$.*

Proof. Assume, without loss of generality, $|GD(u)| \leq |GD(v)|$. Let i_2 be any arc in C_2. Let i_1 be the arc in C_1 whose head is the first one encountered in a counterclockwise traversal from $h(i_2)$. By assumtion, $|GD(i_1)| \leq |GD(i_2)|$.

Claim. $h(i_2) \in (h(LAST(i_1)),h(NEXT(i_1)))$

Proof of Claim. No arc in C_1 can have its head in segment $(h(LAST(i_1)),h(i_1))$ or $(h(i_1),h(i_2))$ by the selection of i_1. Since $h(i_1) \in$ arc $LAST(i_1)$, $h(NEXT(i_1))$ must be encountered after $h(i_2)$ in the clockwise traversal from $h(LAST(i_1))$.

Now, each of the segment $(h(NEXT^{k-1}(i_1)),h(NEXT^k(i_1)))$ for k from 2 until $NEXT^k(i_1) = LAST(i_1)$, can contain at most one of the heads of arcs in C_2, it follows that $|GD(i_2)| \leq |GD(i_1)|$ and these two quantities must be equal. □

By the above analysis, it is easy to verify that the following algorithm correctly determines a good arc in F (it finds the first arc in a directed cycle). We use array L[i] to indicate whether arc i has been visited (when L[i] = 1). An example is shown in Figure 3.1.

Algorithm 1.

```
1. Set $L[i] = 0$, for each arc i;
2. Set $i = 0$;
3. While $L[i] = 0$ do $L[i] = 1$; $i = NEXT[i]$;
4. Report $GD[i]$;
```

4. THE MINIMUM CLIQUE COVER PROBLEM

We shall show that Algorithm 1 can also be used to find a MQC for a proper family F'. Without loss of generality, assume F is not a clique. We need to analyze the structure of F' in more detail.

Definition. A clique is said to be ***linear*** if every arc in the clique passes through a common point of the circle. Denote by **LQ(i)** the linear clique including all arcs passing through t(i) (including arc i itself).

By Lemma 2.6, $\alpha(F') \leq \theta(F') \leq \alpha(F')+1$. Furthermore, if $\theta(F') = \alpha(F')+1$, then there is a minimum clique cover consisting of linear cliques. The following lemma asserts that this is the case even if $\theta(F') = \alpha(F')$.

Lemma 4.2. *If $\theta(F')$ = $\alpha(F')$, and F' is not a clique, then we can find a minimum clique cover for F' consisting of linear cliques.*

Proof. Since θ(F') = α(F'), there exists a collection of cliques C = {Q_1, Q_2, Q_3, ... , Q_k} which cover F'. Let GD(i_1) = {i_1,i_2,i_3,...,i_k} be a maximum independent set of F'. Because F' is not a clique, k must be greater than 1. Hence, each arc of GD(i_1) must be in a different clique of C. Without loss of generality, assume $i_r \in Q_r$. Since NEXT(i_1) does not overlap with any of i_2, i_3, ..., i_k, it must be in Q_1. Hence every arc in Q_1 must overlap with i_1 and NEXT(i_1). Moreover, there cannot exist an arc, say i', in Q_1 which covers the circle together with i_1 and NEXT(i_1), because i' would then properly contain i_k, a contradiction. Hence, Q_1 must be a linear clique. Now, we can apply the above argument to GD(i_2), ..., GD(i_k) to show that Q_2, Q_3, ... ,Q_k are linear clique. □

Every maximal linear clique Q of F' whose arcs pass through a common point p can be written as LQ(i), where i is the arc such that t(i) is the first tail encountered in a counterclockwise traversal from p. Hence, the MQC problem is equivalent to fining a set S = {i_1,i_2,...,i_k} such that the collection of cliques **LQ(S)** = {LQ(i_1),LQ(i_2),...,LQ(i_k)} covers F'. Similar to the NEXT function for maximal independent sets defined in Section 2, we can define, for a given linear clique LQ(i), the **next best** linear clique to be LQ(NEXT(i)). Hence, a minimal greedy clique cover $GD_q(i_1)$ starting from LQ(i_1) can be defined as {LQ(i_1),LQ(i_2),...,LQ(i_k)}, where i_r = NEXT(i_{r-1}) and i_k = LAST(i_1) if LAST(i_1) $\neq$ i_1; otherwise, i_k is chosen so that NEXT(i_k) = LAST(i_1). There is a simple relationship between GD(i) and GD_q(i): if LAST(i) = i then GD_q(i) = GD(i); otherwise GD_q(i) = GD(i) ∪ {LAST(i)}. Define every arcs in a MQC to be a good arc and otherwise, a bad one. Construct the directed graph H the same as before. Similar to those properties of MIS proved in Section 3, one can show that

(4.4) If GC_q(i) is a MQC for F, so is GD_q(NEXT(i)).

(4.5) Let i be any arc in a proper family F'. Let $C_{\min}$ be a MQC for F'. Then $|C_{\min}| \leq |GD(i)| \leq |C_{\min}| + 1$.

(4.5) All arcs in a directed cycle of H are good arcs.

5. THE MINIMUM DOMINATING SET PROBLEM

The MDS problem on circuar graphs was solved by Bonucceli [1] in O(n^2) time by reducing it to problems on interval graphs. Let F' be the proper subfamily of F obtained by deleting all arcs in F which are properly contained in others. For each arc i which is properly contained in an arc j, a dominating set containing i can be changed to another dominating set by swapping i and j. Hence, it suffices to find a

MDS for F from F'. However, bear in mind that a dominating set for F' is not necessarily a dominating set for F.

If there is an arc i ∈ F' which overlaps with every other arc in F, then {i} is a minimum dominating set and $\tau(F) = 1$.. In the following, we assume that $\tau(F) > 1$. To use Algorithm 1, we need to define $NEXT_d(\cdot)$ and $GD_d(\cdot)$ functions which satisfy similar properties as those for $NEXT(\cdot)$ and $GD(\cdot)$ defined in Section 3.

Definition. For each arc i in F', define the ***first clockwise undominated*** FCU(i) arc of i to be the arc u in F\N(i,F) such that t(u) is the first such tail encountered in a clockwise traversal from t(i). Define $NEXT_d(i)$ to be the arc j in N(i,F') such that t(j) is the last such tail encountered in a clockwise traversal from t(u). (there are two cases of $NEXT_d(\cdot)$ as shown in Figure 5.1.

The following properties can be easily justified, where $j = NEXT_d(i)$ and u is the FCU arc of i:

(5.1) Let v be any arc such that t(v) is in segment (t(j), t(i)), then v does not overlap with u (see Figure 5.2).

(5.2) If S is a dominating set of F and $j \notin S$, then there exists at least one arc v in S such that t(v) ∈ segment(t(i), t(j)).

(5.3) If i ∈ F', then j ∈ F'.

(5.4) If there is an arc v such that t(v) ∈ segment(t(i), t(j)), then v is dominated by either i or j.

(5.5) If i belongs to a MDS, then j also belong to some MDS.

Based on $NEXT_d(\cdot)$, we can define $LAST_d(\cdot)$,$GD_d(\cdot)$, and the digraph H_d similarly to those defined in the last section, where $GD_d(i)$ is the minimum greedy dominating set containing i of the following form {i, $NEXT_d(i)$, $NEXT_d^2(i)$, $NEXT_d^3(i)$, ...}, $\tau(F) = \min_i |GD_d(i)|$. And, either $i = LAST_d(i)$ or h(i) ∈ segment ($h(LAST_d(i))$, $h(NEXT_d(LAST_d(i)))$). Define every arc in some MDS of F as a good arc, and others as bad arcs. The proofs in the last section can be easily adapted to show the following properties:

(5.6) Let D_{min} be a MDS for F. Then for any i∈ F', $|D_{min}| \leq |GD_d(i)| \leq |D_{min}|+1$.

(5.7) All arcs in a directed cycle of H_d are good arcs.

Therefore, Algorithm 1 can be used to find a MDS for F. However, unlike the MIS or the MQC problem, where one does not have to precompute NEXT(i) for every arc i in F' (only compute them when encountered in Algorithm 1), it is necessary for us to perform this preprocessing step for $NEXT_d(\cdot)$ because it is more involved. Below, we describe an O(n) time algorithm which computes $NEXT_d(i)$ for each arc i in F'.

Let $i_1, i_2, \ldots ,i_r$ be the arcs in F' arranged in clockwise order starting with i_1.

Our algorithm calculates NEXT(i) by first finding FCU(i) in the following order: FCU(i_1), NEXT(i_1), FCU(i_2), NEXT(i_2), ..., FCU(i_r), NEXT(i_r). The following lemma states that the tails of the FCU(i)'s are clockwise ordered and so does the NEXT(i) sequence.

Lemma 5.8. *For any i, i' ∈ F', segment* $(t(i),t(u))$ *is not properly contained in segment* $(t(i'),t(u'))$*, and segment* $(t(i),t(j))$ *is not properly contained in segment* $(t(i'),t(j'))$ (See Figure 5.3).

Algorithm 2. (To calculate $\mathrm{NEXT}_d(\cdot)$ function)

```
1.  Sort all arcs in F according to their tail orders. Construct a circular linked list
    P on arcs in F, where P(i) is the first clockwise tail after t(i) in F, and another
    linked list P' on arcs in F', where P'(i) is the first clockwise tail after t(i) in F'.
2.  Arbitrarily select an arc i* in F'. If N(i*,F) = F, then stop, ({i*} is the
    minimum dominating set). Otherwise, let arc u* = FCU(i*) and j* =
    NEXT_d(i*). Let i = P'(i*), u = u*, j = j*.
3.  While (i ≠ i*), repeat steps 4 to 8.
4.  If u = i then u ← P(u)
5.  While (u ∈ N(i,F) and u ≠ i) do u ← P(u)
6.  If i = u then stop; ({a} is a minimum dominating set)
7.  While (P'(j) contain t(u)) do j ← P'(j);
8.  NEXT_d(i) = j
```

It can be easily checked that Algorithm 2 also tests whether there is an arc which dominates all arcs in F. Step 4 guarantees that i ≠ u at the outset of Step 5. Then, if there is an arc i dominating all others in F, the loop in Step 5 will end with i = u. Finally, the algorithm will terminate in Step 6 with $\tau(F) = 1$. From Lemma 5.8, it is easy to chek that Algorithm 2 can be executed in O(n) space and O(n) time.

REFERENCES

1. Bonuccelli, *Dominating sets and domatic numbers of circular–arc graphs*, **Discrete Appl. Math.** 12 (1985), pp. 203–213.
2. U.I. Gupta, D.T. Lee and J. Y.–T. Leung, *Efficient algorithms for interval graphs and circular–arc graphs*, **Networks**, 12 (1982), pp. 459–467.
3. W.–L. Hsu, *A simple linear time algorithm for the maximum independent set problem on circular–arc graphs*, (1988), submitted.
4. S. Masuda and K. Nakajima, *An optimal algorithm for finding a maximum independent set of a circular–arc graph*, **SIAM J. Comput.**, 17 (1988), pp. 41–52.

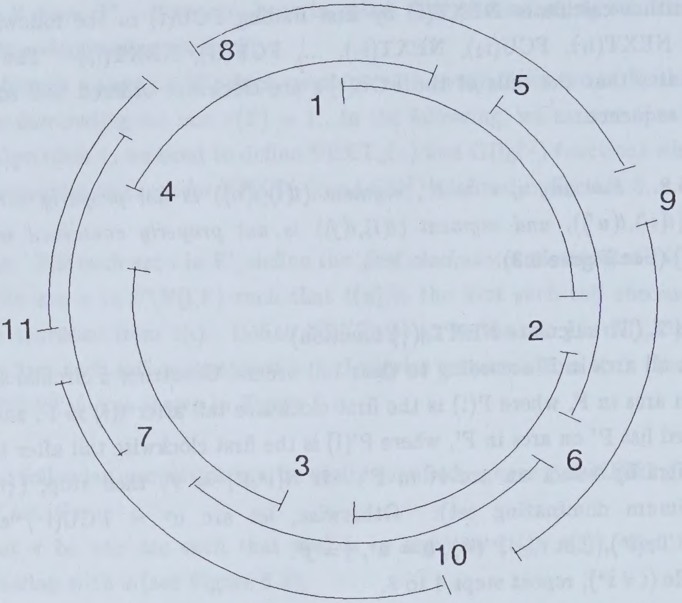
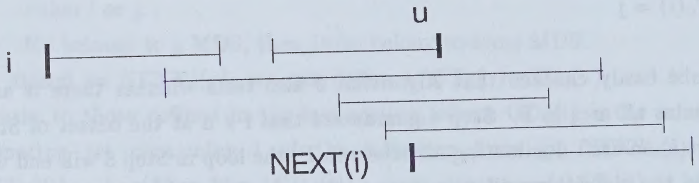
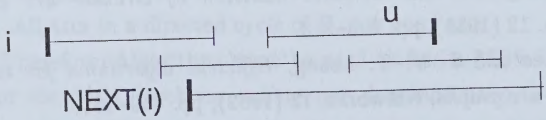

8
5
1
4
9
11
2
7
6
3
10

Figure 3.1



(a) i, NEXT(i) do not overlap.



(b) i, NEXT(i) overlap.

Figure 5.1

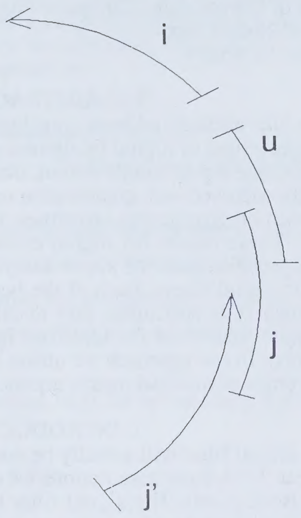


Figure 5.2

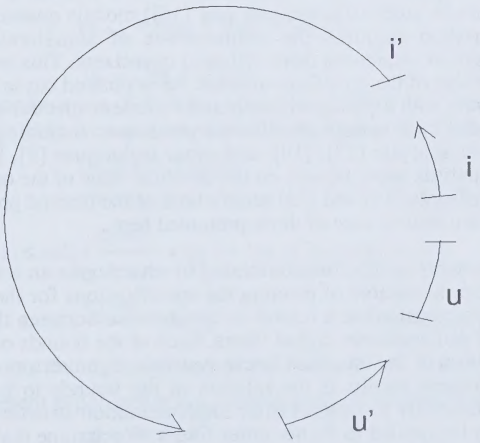


Figure 5.3

NEW ESTIMATES OF ASYMPTOTIC TRAJECTORY BOUNDS FOR DIGITAL FILTERS: IMPROVED RESULTS*

J. A. FARRELL AND A. N. MICHEL
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, In. 46556

ABSTRACT

Recent results by the authors address qualitative properties (e.g., uniform ultimate boundedness of trajectories) of digital implementations of feedback controllers (i.e., linear, finite dimensional, single input - single output, time-invariant plants compensated by digital controllers which are endowed with quantization nonlinearities). These results were extended by [14] which focused on quantitative properties of digital control systems (e.g. estimates of trajectory bounds). These results for digital control systems are also applicable to digital filters. The present paper extends the above analysis by presenting a new class of trajectory bound estimates for digital filters. Each of the bounds in this class is applicable to general digital filter implementation structures, easy to calculate, and dependent on parameters(i.e., eigenvalues and eigenvectors) of the idealized linear system (i.e. the digital filter model neglecting quantizers). In our approach we utilize the comparison principle in the Lyapunov theory and the properties of minimal matrix norms.

I. INTRODUCTION

Although a digital filter will actually be nonlinear, for design purposes it will usually be modelled as linear. With these assumptions we are able to use the numerous linear system design tools (e.g., Bode plots). The digital filter is, however, inherently nonlinear. This is due to the fact that computers store numbers with only a finite number of bits; hence, quantization and overflow nonlinearities are unavoidable. These nonlinearities are known to cause oscillation in the states of digital filters. A looming research topic is the determination of quantization oscillation bounds which are applicable to general filter implementations and computable for higher order filters.

Past work on quantization effects has benefitted from a variety of different viewpoints. For example, stochastic analysis (see [16]) models quantization errors as noise sequences. This method requires the assumptions of statistically independent and uncorrelated quantization sequences from different quantizers. This method yields bounds on the mean square value of the quantization noise. As is pointed out in [17] these bounds are accurate in applications with high signal levels and sufficient spectral content. Deterministic analysis has proceeded by a variety of different techniques including: Lyapunov analysis [12], [17]; limit cycle analysis [17], [19]; and other techniques [9], [10], [13], [15], [18]. Most deterministic analysis seeks bounds on the absolute value of the output and states of the digital filter. However, only [13] and [15] satisfy both of the desired properties stated above. The results in [13] are a special case of those presented here.

This paper presents results for determining to what degree an actual digital filter, with its inherent quantizers, is capable of meeting the specifications for the idealized filter. The method we propose is to calculate a bound on the distance between the trajectories of the corresponding linear and nonlinear digital filters. Each of the bounds on this distance, which we present, is a function of the idealized linear system's eigenvectors and eigenvalues. The uniqueness of the present results is the relation of the bounds to the idealized system's parameters, the applicability to general filter implementation structures, and the ease with which the bound can be applied to higher order filters. We assume that the filter is properly scaled so that overflow does not occur.

---

*This work was supported in part by the National Science Foundation under grant No. Ecs .88-02924, and by the University of Notre Dame Applied Mathematics Center.

II. NOTATION AND PRELIMINARIES

The present section formalizes the questions which this paper addresses and presents the notation and concepts used in the remainder of this paper.

PART A- QUANTIZERS. The results of the present paper apply to both fixed and floating point arithmetic. Although there are a number of quantization types (i..e. rounding, two's complement truncation, and one's complement truncation), all these various quantization functions, Q(x), can be decomposed as

$$Q(x) = x + p(x) \text{ for } x \in R$$

where $$|p(x)| < \varepsilon \text{ for fixed point arithmetic} \tag{1a}$$

and $$|p(x)| < \varepsilon_1|x| + \varepsilon_2 \text{ for floating point arithmetic.} \tag{1b}$$

The ε in (1a) depends on both the type of quantization and the number of bits used to represent x . In (1b), ε_1 is dependent on the number of bits representing the mantissa, while ε_2 depends on the number of bits representing the characteristic. Both ε_1 and ε_2 depend on the type of quantization used. For further discussion of the quantization nonlinearity see [1, ch.9].

PART B- MODELS. Given a transfer function H(z) for the idealized digital filter, let a state space realization be described by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{2}$$

where $A \in R^{n \times n}$, $x : I \rightarrow R^n$, $B \in R^n$, $C \in R^n$, $y : I \rightarrow R$, $u : I \rightarrow R$, and I denotes the set of nonnegative integers. Of course, this realization is not unique, but the bounds to be derived will apply to any such realization of H(z).

The actual digital filter corresponding to (2) will be described by

$$\begin{aligned} p_{k+1} &= Ap_k + Bu_k + R1(k) \\ v_k &= Cp_k + R2(k) \end{aligned} \tag{3}$$

where R1(k) is a vector function of both p_k and u_k, and R2(k) is a real valued function of p_k. As was presented in [14], we can find positive real constants such that

$$| R1_i(k) | \leq n1_i \, \varepsilon$$

and $$| R2(k)| \leq n2 \, \varepsilon \qquad \text{for fixed point arithmetic,}$$

also $$| R1_i(k) | \leq m1_i \left(\frac{\varepsilon_1 \|x\|}{\sigma} + \varepsilon_2 \right) + | h_i | (\varepsilon_1 | u(k) | + \varepsilon_2)$$

and $$| R2(k) | \leq m2 \left(\frac{\varepsilon_1 \|x\|}{\sigma} + \varepsilon_2 \right) \qquad \text{for floating point arithmetic.}$$

In the above $R1_i$ denotes the ith coordinate of the vector R1, $\| . \|$ is the norm which is the main subject of the present paper, and σ is a positive constant satisfying

$$\sigma | x_i | \leq \| x \|$$

for any $x = (x_1, ... ,x_n)$, and $i = 1, ... ,n$. For later use in this paper, define

$$N1 = (n1_1, ... ,n1_n)^T$$

and $$M1 = (m1_1, ... ,m1_n)^T.$$

At this point it is helpful to note that the constants n1 and m1 are dependent on the actual implementation, and the filter parameters. However, the constants are easily determined.

These constants do affect the trajectory bounds; hence, a computer program could compare alternate implementation structures on the basis of the trajectory bound.

III. HOUSEHOLDER NORMS

This section presents results from [8] which are essential for the understanding of the proofs in section V. The results presented here are not meant to summarize [8]; hence, reference [8] should be consulted for further results on the topics of this section.

Let K be a subset of a real or complex linear space U. Recall that K is called convex when $x, y \in K$ implies that $\lambda x + (1-\lambda)y \in K$, for any $\lambda \in [0,1]$. In addition, K is said to be equilibrated if $x \in K$ implies $wx \in K$ when $|w| < 1$.

Result 1- If K is complex and equilibrated, then

1) $0 \in$ interior of K, and

2) for any $x \in U$, there exists $\alpha \in R$ such that $x \in \alpha K$

where $\alpha K = \{x \in U \mid x = \alpha v, v \in K\}$.

Item one above can be used to show that for convex and equilibrated K,

$$\| x \|_K = \text{glb}\,(v \in R \mid v \geq 0,\ x \in vK)$$

is a norm on U. If $A \in R^{n\times n}$, and if A maps U into U, then define

$$\text{lub}(A)_K = \text{glb}\,(\alpha \in R \mid \alpha \geq 0, \alpha K \supset AK).$$

The function $\| x \|_K$ will be called the norm of x with respect to K. With these definitions it is easy to show that

$$\| Ax \|_K \leq \| x \|_K\, \text{lub}(A)_K.$$

Result 2- If P is a nonsingular nxn matrix and K is a convex, equilibrated region, then

1) $H = PK$ is convex and equilibrated

and 2) $\| x \|_H = \| P^{-1}x \|_K$.

Result 3- Under the assumptions of Result 2, if A is an nxn matrix, then

$$\text{lub}(A)_H = \text{lub}(P^{-1}AP)_K.$$

The importance of Results 2 and 3 will be in showing that the vector norm and matrix bound are invariant under a similarity transformation when the underlying region is transformed in the same manner as the vector. By these results, we are able to analyze in place of the state transition matrix A, any similar matrix, $S = T^{-1}AT$.

In addition to the above results, we will need the following specific results for our subsequent analysis. The p-norm of a vector in the linear space U is defined by

$$\| x \|_p = \left(\sum_{i=1}^{n} | x_i |^p \right)^{1/p}$$

for $1 \leq p \leq \infty$. The unit ball with respect to each p-norm is defined by

$$B_p = \{ v \in U \mid \| v \|_p \leq 1 \}.$$

For the values of p = 1, ∞ the corresponding matrix norm is particularly easy to calculate:

$$\| A \|_{B1} = \max_i \left(\sum_{k=1}^{n} | a_{ki} | \right)$$

and $$\| A \|_{B\infty} = \max_i \left(\sum_{k=1}^{n} | a_{ik} | \right).$$

For the special case in which A is a diagonal matrix, the p = 2 norm is also easy to calculate. In fact,

$$\| A \|_{B1} = \| A \|_{B\infty} = \| A \|_{B2} = \rho(A)$$

where $\rho(A)$ is the magnitude of the largest eigenvalue (or spectral norm) of A.

IV. PRELIMINARY RESULTS

This section presents two theorems which form the foundation for the remainder of this paper. These and other related results are proved in [2] . Before presenting these theorems, we need the following definitions.

Definition 1. The linear system (2) with u(k) = 0 is said to be uniformly asymptotically stable if and only if all eigenvalues of the matrix A satisfy $| \lambda_i | < 1$.

Definition 2. The nonlinear system (3) with u(k) = 0 is said to be uniformly ultimately bounded with bound α if and only if for any B > 0 there exists T(B) > 0, independent of K > 0, such that whenever $| p(K) | \leq B$ and $k \geq T(B)$, one has $| p(K+k) | \leq \alpha$.

In terms of the systems discussed in Section II, [4] proved the following two theorems.

Theorem 1- In (3) assume that A is a stable matrix and that there exist constants $n_i \geq 0$ such that $| R_i(k) | < n_i\varepsilon$ (denote this as $|R(k)| < N\varepsilon$, $N = (n_1, \dots, n_{n+m})$). Then there exists a norm $\| \cdot \|$ and a constant $0 < \varsigma < 1$, both of which depend only on the matrix A, such that the solutions p and x of (3) and (2) respectively, satisfy

$$\| p(k) - x(k) \| \leq \frac{\| N \| \varepsilon}{1 - \varsigma} + B(k),$$

where $$B(k) = \left[-\frac{\| N \|}{1 - \varsigma} \varepsilon + \| p(0) - x(0) \| \right] \varsigma^k \to 0 \quad \text{as } k \to \infty.$$

Remark 1. Note that the assumptions of this theorem are satisfied if (2) is designed to be asymptotically stable and fixed point arithmetic is used.

Remark 2. This theorem does not assume that the input $r \equiv 0$; thus, the theorem states that for any input, the distance between the idealized system and the nonlinear system states will be less than the given bound. Also, if the input is identically zero then x(k) approaches zero; the theorem then implies that

$$\| p(k) \| < \frac{\| N \|}{1 - \varsigma} \varepsilon \qquad \text{as } k \to \infty.$$

Theorem 2- Assume that A in (3) is a stable matrix and that there exist constants m_i such that

$$R_i(k) \leq m_i\left(\frac{\varepsilon_1}{\sigma} \| p(k) \| + \varepsilon_2 \right) + | h_i |\left(\varepsilon_1 | u(k) | + \varepsilon_2 \right)$$

for all $p(k) \varepsilon R^{n+m}$, $u(k) \varepsilon R$ (denote this by $| R(k) | \leq M(\frac{\varepsilon_1}{\sigma} \| p(k) \| + \varepsilon_2) + | h |(\varepsilon_1 | u | + \varepsilon_2)$, where $M = (m_1, \dots, m_{m+n})$ and $| h | = (| h_1 |, \dots, | h_{n+m} |)$). Then, there exists a norm $\| \cdot \|$ and a constant $0 < \varsigma < 1$ both of which depend only on the matrix A such that if ε_1 is small enough so that $0 < \Sigma < 1$ then the solutions p and x of (3) and (2), respectively, satisfy

$$\| p(k) - x(k) \| \leq \text{Ç}^{k} \| p(0) - x(0) \| + \left(\frac{1 - \text{Ç}^{k}}{1 - \text{Ç}} \right) \varepsilon_2 (\| M \| + \| h \|)$$

$$+ \sum_{j=0}^{k} \varepsilon_1 \text{Ç}^{k-j} (\| M \| \| x \| + \| h \| |u(k)|)$$

or $$\| p(k) - x(k) \| \leq \left(\frac{1}{1 - \text{Ç}} \right) \varepsilon_2 (\| M \| + \| h \|)$$

$$+ \sum_{j=0}^{k} \varepsilon_1 \text{Ç}^{k-j} (\| M \| \| x \| + \| h \| |u(k)|)$$

as $k \to \infty$, where $\text{Ç} = (\text{ç} + \frac{\varepsilon_1}{\sigma} (\| M \| + \| h \| \| \tilde{F} \|))$, and

$$\tilde{F} = [\, F^{T} \; 0 \; \dots \; 0 \,]^{T}.$$

Remark 3. The assumptions of this theorem will be satisfied when (2) is designed to be asymptotically stable and floating point arithmetic is used.

Remark 4. The bound of Theorem 2 is more complex than that of Theorem 1. The bounds are still both simple to calculate. The norm $\| \cdot \|$ and constant ç are the only unknown quantities. These unknowns are the subject of this paper, and their determination will be addressed in Section V.

Remark 5. The exact value of Theorem 2's bound is dependent on the idealized system's steady state tracking error. As in Remark 2, if the input is identically zero, we have

$$\| p(k) \| \leq \frac{\varepsilon_2 (\| M \| + \| h \|)}{1 - \text{Ç}} \qquad \text{as } k \to \infty.$$

The two previous theorems are applicable to digital filters. As was presented in [2] this type of analysis is also applicable to digital control systems at the sampling instants. Two theorems first presented in [2], present bounds applicable to the continuous time variables as well.

V. MAIN RESULT

Section IV discussed two theorems which yield bounds on the error in the state of a nonlinear digital filter. These bounds contain certain parameters which depend on the particular system and topology of interest. All of these parameters are known and easy to calculate except for the unknown ç and the norm $\| \cdot \|$. In [2] and [4] we stated that this norm was known to exist and that the required constant could be found by a variety of methods, including the method of Brayton and Tong [5], [6].

In order to calculate quantitative bounds we must know the actual form of the norm, $\| \cdot \|$. In general, this norm is not a p-norm (see Section III). In this section we present a class of norms which satisfy the required conditions in Theorems 1 and 2. We also show that the required constant ç is the spectral norm of the matrix A. Thus this paper establishes a relationship between an important parameter of the idealized system, the spectral norm of A, and an important parameter of the actual nonlinear system, the trajectory bound.

The main result of this paper is presented in Theorem 3. The proof of this theorem is completed in two parts. We will only present the proof for the case of distinct eigenvalues. The proof for the case of repeated eigenvalues is similar to that given in [14] for the p=2 norm.

Theorem 3- If the algebraic and geometric multiplicities are equal for each eigenvalue of A which has magnitude equal to $\rho(A)$, then the bounds of Theorems 1 and 2 can be calculated using the norm

$$\| x \| = \| Px \|_p$$

and the constant $\text{ç} = \rho(A)$,

where $1 \le p \le \infty$.

Remark 6. The matrix P may have one of two forms depending on the eigenstructure of A. As will be shown for the case of distinct eigenvalues in the proof to follow, if A is similar to a diagonal matrix D ($A = TDT^{-1}$) then $P = T^{-1}$. This is true, for example, when the Jordan form of A is a diagonal matrix, in which case T is the matrix whose columns are the eigenvectors of A. If the matrix A is not diagonalizable, then $P = (WT^{-1})$, where the form of the matrix W is mentioned in [8] and [14].

Remark 7. By specifying the form of the norm, $\| \cdot \|$, and the constant, ç, Theorem 3 shows that for fixed point arithmetic the bound on the norm of the error due to quantization is given by Theorem 1 with $\| e(k) \| = \| x(k) - p(k) \|$. Also, the bound on the norm of the error due to quantization for floating point quantization is given by theorem 2. The bound on the norm can be translated into a bound on the absolute value of the error in the ith component of the state. This will be discussed following the proof.

Remark 8. In order to prove the theorem, it is necessary to show that the norm and constant given in Theorem 3 satisfy the conditions necessary to complete the proofs of Theorems 2 and 3. Specifically, the condition

$$\| Ax \| \le \rho(A) \| x \|$$

must be satisfied for all $x \in R^n$. The condition, $0 < \rho(A) < 1$, is automatically satisfied by asymptotically stable systems described by (2).

PROOF: CASE ONE- DISTINCT EIGENVALUES

For each eigenvalue $\lambda_i(A)$, let v_i denote the associated eigenvector. Define

$$T = [\, v_1 , \dots , v_n \,],$$

so that

$$S = T^{-1}AT = \mathrm{diag}(\lambda_i)$$

Let $v = (\zeta_1 , \zeta_2 , \dots , \zeta_n)^T = T^{-1}x$, $\quad x \in R^n$, and $\zeta_i \in C$. Then

$$Sv = (\zeta_1\lambda_1, \zeta_2\lambda_2 , \dots , \zeta_n \lambda_n)^T$$

and

$$\| Sv \|_p = \Big(\sum_{i=1}^{n} |\zeta_i\lambda_i|^p \Big)^{1/p} \le \rho(A) \Big(\sum_{i=1}^{n} |\zeta_i|^p \Big)^{1/p} = \rho(A) \| v \|_p .$$

By results 2 and 3 of Section III,

$$\| Ax \|_{TB_p} = \| TST^{-1}x \|_{TB_p} = \| Sv \|_{B_p} \le \rho(A) \| v \|_{B_p} = \rho(A) \| x \|_{TB_p}$$

where $\| \cdot \|_p = \| \cdot \|_{B_p}$, and $B_p = \{ v \mid \| v \|_p \le 1 \}$. This proves case 1 of Theorem 3 with

$$\| x \| = \| x \|_{TB_p} = \| T^{-1}x \|_{B_p} = \| Px \|_p .$$

Hence, for case 1, $P = T^{-1}$, where T is the matrix of eigenvectors of A.

Remark 9. Although $\| v \|_{B_p}$ corresponds to a p-norm, the norm of interest, $\|x\|_{TB_p}$, does not, in general, correspond to any p-norm. The unit sphere under $\| x \|_{TB}$ will depend on the eigenvectors of the matrix A.

With $\text{ç} = \rho(A)$, the final questions of interest are: what are the easiest and least conservative methods to calculate $\|N\|$, $\|M\|$, and $\|h\|$; and, are there any remaining degrees of freedom which can be used to decrease the conservatism of these bounds? Before proceeding to address these questions, we notice that h is a vector while M and N are the bounds on the extreme points of the region bounding the perturbation R. Thus, we must calculate the norm of h and bounds for M and N.

Let L be the nonsingular transformation that is of interest, either $L = T^{-1}$ (for case 1) or $L = WT^{-1}$ (for case 2). Then

$$v = Lx \text{ and } \| v \|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} = \| Lx \|_p.$$

Thus, $\| x \| = \| Px \|_p$.
Since h is a vector, $\| h \|$ can be calculated directly.

Consider calculating the bound on M; the calculation for N is similar. Each component m_i of M bounds the perturbation to the ith component w_i of w. Thus, M defines a rectangular, equilibrated region $\breve{S}$ about the origin. In order to bound M we must find max($\|x\| \mid x \,\varepsilon\, \breve{S}$). Due to the fact that $\breve{S}$ is an equilibrated polytope, we only need to bound the extreme points of $\breve{S}$ (i.e., $(\pm m_1, \ldots, \pm m_n)$). This requires 2^{n-1} calculations of norms of the extreme points of $\breve{S}$.

At this point the bound is completely known. The last issue to address is conservatism. The remaining available degrees of freedom are the choice of the size of the diagonal elements in the blocks forming W (as long as each ε is less than $\rho(A)$), and the freedom involved in choosing the magnitude of the eigenvectors of A. Using these degrees of freedom, the shape of the unit circle under $\| \cdot \|$ can be manipulated to better apply to specific problems; thus decreasing the bound.

Theorem 3 yields a bound on the norm of the error. If the bound on the absolute value of a particular component of the error vector is of interest, this information is easily obtained from the norm. Let

$$u^{(i)T} = e^{(i)T}T$$

where $e^{(i)}$ is the ith coordinate vector (the ith row of an $n \times n$ identity matrix). Then $u^{(i)T}$ is the ith row of T. Also, since

$$x = Tv,$$

we have that

$$x_i = e^{(i)T}Tv = u^{(i)T}\, v. \tag{4}$$

With the required condition that

$$\| v \|_p \le \frac{\| PN \|_p \varepsilon}{1 - \rho(A)}, \tag{5}$$

the ith coordinate of x will be largest when v is in the direction of $u^{(i)}$. Hence, to satisfy (5) let

$$v = \frac{u^{(i)}}{\| u^{(i)} \|_p} \left(\frac{\| PN \|_p \varepsilon}{1 - \rho(A)} \right).$$

Then from (4) we obtain

$$| x_i | \le \frac{\| u^{(i)} \|_2{}^2}{\| u^{(i)} \|_p} \left(\frac{\| PN \|_p \varepsilon}{1 - \rho(A)} \right). \tag{6}$$

VI. EXAMPLE

To illustrate the use of Theorem 3 consider a second order, direct form digital filter (see figure 1). For the transfer function

$$H(z) = \frac{1}{1 + a\, z^{-1} + b\, z^{-2}},$$

the corresponding state transition matrix is

$$A = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix}.$$

Denoting the eigenvalues by λ_1 and λ_2, the eigenvectors are, respectively,

$$v_1 = \begin{bmatrix} 1 \\ \lambda_1 \end{bmatrix} \text{ and } v_2 = \begin{bmatrix} 1 \\ \lambda_2 \end{bmatrix};$$

hence,
$$T = \begin{bmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{bmatrix} \quad \text{and} \quad T^{-1} = \frac{\begin{bmatrix} \lambda_2 & -1 \\ -\lambda_1 & 1 \end{bmatrix}}{(\lambda_1 - \lambda_2)},$$

for the case of distinct eigenvalues. By the use of (6) with $p = 1, 2, \infty$ we find that either

$$|x_i| \le \frac{(\lambda_1{}^*\lambda_1 + \lambda_2{}^*\lambda_2)}{(|\lambda_1| + |\lambda_2|)} \frac{2}{|\lambda_1 - \lambda_2|} \frac{\varepsilon}{(1 - \rho(A))} \qquad \text{for } p=1, \tag{7}$$

$$|x_i| \le \frac{\sqrt{2}}{|\lambda_1 - \lambda_2|} \frac{\varepsilon \sqrt{\lambda_1{}^*\lambda_1 + \lambda_2{}^*\lambda_2}}{(1 - \rho(A))} \qquad \text{for } p=2, \tag{8}$$

or
$$|x_i| \le \frac{(\lambda_1{}^*\lambda_1 + \lambda_2{}^*\lambda_2)}{\rho(A)} \frac{1}{|\lambda_1 - \lambda_2|} \frac{\varepsilon}{(1 - \rho(A))} \qquad \text{for } p=\infty, \tag{9}$$

where i = 1 and 2. These calculations assume that $N = [\,0\,,\,1\,]^T$, which is equivalent to assuming a quantizer as indicated by the x on figure 1. The calculations also assume that the input is zero so that analog to digital conversion errors can be ignored.

For a specific example, we choose our parameters so that

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0.9 \end{bmatrix}.$$

Thus, equations (7), (8), and (9) yield, respectively,

$$|x_i| \le 20\varepsilon = 0.1$$

$$|x_i| \le 14\varepsilon = 0.07$$

$$|x_i| \le 10\varepsilon = 0.05.$$

For this example, round-off quantization with a step size of 0.01 is used; hence, $\varepsilon = 0.005$. Figure 3 shows the absolute value of the error between the second states of the linear and nonlinear systems. As this figure shows, the bound using $p = \infty$ or $\|x\| = \|x\|_{T^{-1}B_\infty}$ is exact. The resulting regions $T^{-1}B_1$, $T^{-1}B_2$, and $T^{-1}B_\infty$ are each shown in figure 2.

REFERENCES

[1] Oppenhiem, A. V. and R. W, Schafer, Digital Signal Processing, New Jersey, Prentice-Hall, 1975.

[2] Miller, R.K., M.S. Mousa, and A. N. Michel, "On Dynamic Effects of Quantization and Overflow Nonlinearities in Digital Feedback Control Systems," IEEE Trans. on Autom. Control, Vol. AC-33, No 7, pp. 698-704, July 1988.

[3] Sandberg, I. W., "Floating-Point-Roundoff Accumulation in Digital Filter Realizations," Bell System Technical Journal, Vol. 46, pp. 1775-1791, October 1967.

[4] Miller, R. K., A. N. Michel, and J. A. Farrell, "Quantizer Effects on Steady-State Error Specifications of Feedback Control Systems," to appear in IEEE Trans. on Automatic Control, vol. AC-34, no. 4, April 1898.

[5] Brayton, R. K. and C. H. Tong, "Stability of Dynamical Systems: A Constructive Approach," IEEE Transactions on Circuits and Systems, Vol. CAS -26, No.4, pp. 224-234, April 1976.

[6] Brayton, R. K. and C, H, Tong," Constructive Stability and Asymptotic Stability of Dynamical Systems, " IEEE Transactions on Circuits and Systems, Vol. CAS-27, No. 11, pp. 1121-1130, November 1980.

[7] Miller, R. K. and A. N. Michel, Ordinary Differential Equations, New York, Academic Press, 1982.

[8] Householder, A. S., The Theory of Matrices in Numerical Analysis, New York, Blaisdell, 1964.

[9] Jackson, L. B., "An Analysis of Limit Cycles Due to Multiplication Rounding in Recursive Digital (Sub) Filters," Proc. Seventh Annual Allerton Conf. on Circuit and System Theory, Monticello, Illinois, pp. 69-79, October 1969.

[10] Bertram, J. E., "The Effect of Quantization in Sampled Feedback Systems," Transactions AIEE Appl. Ind., Pt. 2, Vol. 77, pp. 177-181, September 1958.

[11] Slaughter, J. B., "Quantization Errors in Digital Control Systems," IEEE Trans. on Automatic Control, Vol. AC-9, pp. 70-74, January 1964.

[12] Johnson, G. W., "Upper Bound on Dynamic Quantization Error in Digital Control Systems Via the Direct Method of Liapunov," IEEE Trans. on Automatic Control, Vol. AC-10, No. 4, pp. 439-448, October 1965.

[13] Green, B.D. and L. E. Turner, "New Limit Cycle Bounds For Digital Filters," IEEE Transactions on Circuits and Systems, Vol. CAS-35,No. 4, pp. 365-375, April 1988.

[14] Farrell, J. A. and A. N. Michel, "Estimates of Asymptotic Trajectory Bounds in Digital Implementations of Linear Feedback Control Systems," Proceedings of the 27th IEEE Conference On Decision And Control, Austin, Texas, December 7-9, 1988.

[15] Yakowitz, S. and S. R. Parker, "Computation of Bounds for Quantization Errors, "IEEE Transactions on Circuit Theory, vol. 20, no. 4, July, 1973, pp. 391-396.

[16] Bede Liu, " Effect of Finite Word Length on the Accuracy of Digital Fiters- A Review, "IEEE Transactions on Circuit Theory, vol. 18, no. 6, November, 1971, pp. 670-677.

[17] Parker, S. R. and S. F. Hess, "Limit-Cycle Oscillations in Digital Filters," IEEE Transactions on Circuit Theory, Vol. CT-18, No. 6, pp. 687-697, November 1971.

[18] Sandberg, I. W. and J. F. Kaiser, "A Bound on Limit Cycles in Fixed-Point Implementations of Digital Filters,"IEEE Trans. on Audio and Electroacoustics, Vol. AU-20, No. 2, PP. 110-112, June 1972.

[19] Long, J. L. and T. N. Trick, "An Absolute Bound on Limit Cycles Due to Roundoff Errors in Digital Filters," IEEE Trans. on Audio and Electroacoustics, Vol. AU-21, No.1, pp. 27-30, February 1973.

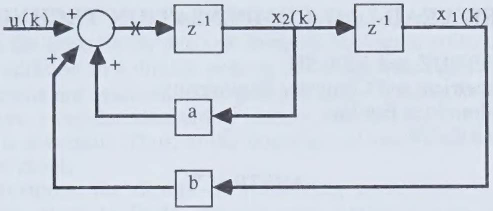


Figure 1- Linear second order direct form filter.

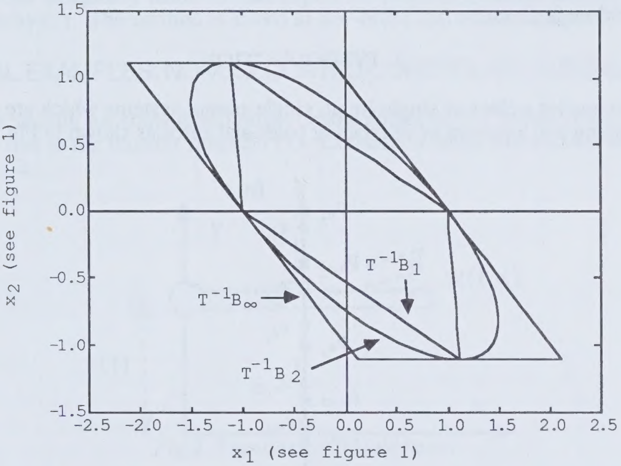


Figure 2- Regions in the state space of the direct form filter (x space) for the example which define norms equivelent to the p= 1, 2, and infinity norms in the v space of Theorem 3's proof.

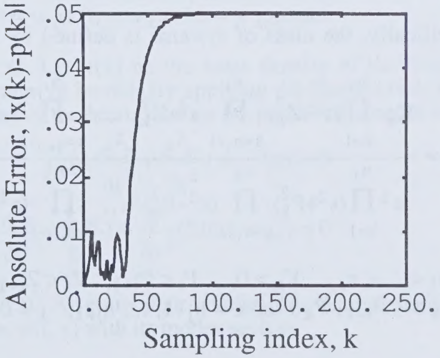


Figure 3- Absolute value of the error between the first state of the linear system and the first state of the non-linear system versus the sampling index.

A GENERALIZED LEAD-LAG COMPENSATION TECHNIQUE

NASSIM M. ARBOUZ and REN SU
Department of Electrical and Computer Engineering
University of Colorado at Boulder
Boulder, CO 80309-425

ABSTRACT

We present a controller design technique for a class of systems which are characterized by negligible structural damping and a pattern of interlacing poles and zeros. The goal is to achieve robust low-order compensation for high-order plants. The technique, based on the root-locus method, is a generalization of the classical lead-lag compensation. A theorem on the existence of such a stabilizing controller is proved and a systematic design procedure is given.

I. INTRODUCTION

We consider a class of single-input, single-output systems which are characterized by zero damping and a pattern of interlacing poles and zeros as shown in Fig. 1.

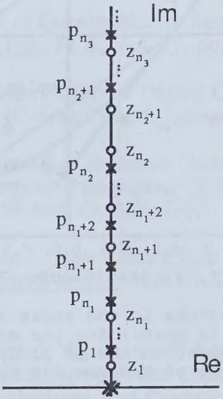


Fig. 1. Plant pole-zero locus

More specifically, the class of systems is defined by transfer functions of the following form

$$G(s) = \frac{K_g \prod_{k=1}^{n_1}(s^2+Z_k^2) \prod_{k=n_1+1}^{n_2} (s^2+Z_k^2) \ \dots \ \prod_{k=n_{r-1}+1}^{n_r} (s^2+Z_k^2)}{s^2 \prod_{k=1}^{n_1}(s^2+P_k^2) \prod_{k=n_1+1}^{n_2} (s^2+P_k^2) \ \dots \ \prod_{k=n_{r-1}+1}^{n_r} (s^2+P_k^2)} \tag{1}$$

such that $0 < n_1 < \dots < n_r$, $K_g > 0$, $P_i < P_{i+1}$, $Z_i < Z_{i+1}$ for $i = 0, \dots, n_{r-1}$, and $Z_k \in (P_{k-1}, P_k)$ for $k = n_j+1, \dots, n_{j+1}$; $j = 0, 2, 4, \dots$

$$Z_k \in (P_k, P_{k+1}) \text{ for } k = n_j+1, \ldots, n_{j+1}; \quad j = 1, 3, 5, \ldots$$

A system in the class has n_r pairs of complex conjugate poles and zeros, all on the imaginary axis, in addition to a double pole at the origin (the rigid body mode). The n_r pairs of poles and zeros are assumed to be grouped into r subsets. Within each subset the poles and zeros have a regular alternating pattern, but from one subset to another, the alternating pattern is reversed. Thus, at the boundary of two subsets there are either two consecutive poles or zeros.

For these systems, the design of stabilizing controllers is not an easy task. Classical techniques, such as Bode, are not very useful due to the irregularities in interlacing. One might use state-variable methods to produce a state feedback - observer type of controller. However, to ensure stability, the controller could be of very high order, therefore very sensitive to modeling uncertainties. The $H\infty$ approach to robust control may be one possible area of investigation, although some methods [3] do not handle plant poles and zeros on the imaginary axis. In this paper, we propose a compensation scheme for systems given by (1). The method is aimed at low-order and robust designs.

II. PHYSICAL EXAMPLES: NONCOLOCATED CONTROL OF FLEXIBLE BEAMS

In this section we show that flexible beams with a pair of noncolocated sensor and actuator give rise to the transfer function (1). Consider a beam in translational motion as shown in Fig. 2.

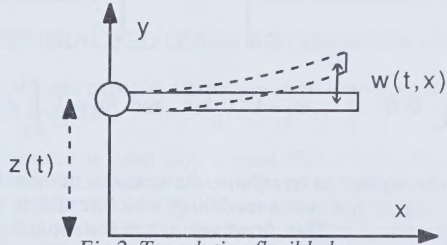


Fig 2. Translating flexible beam

The beam has length L and moves in the (x, y) plane with one end (x = 0) clamped to an actuator and another end (x = L) free. The displacement of the beam from y = 0 at x = 0 is denoted by z(t), and the flexure of the beam with reference to y = z(t) is denoted w(t, x). This configuration amounts to the following boundary conditions on w(t, x)

$$w(t,0) = w_x(t,0) = w_{xx}(t,L) = w_{xxx}(t,L). \tag{2}$$

Let v(t, x) = z(t) + w(t, x). Let $\rho(x)$ be the mass density of the beam, E the Young's modulus, and I(x) its moment of inertia. By applying the Hamiltonian principle, it can be shown [1] that the dynamics of the beam satisfies the partial differential equation

$$\int_0^L \rho.\left(\frac{d^2 z}{dt^2} + \frac{\delta^2 w}{\delta t^2}\right) dx - f(t) = 0$$

$$\rho.\left(\frac{d^2 z}{dt^2} + \frac{\delta^2 w}{\delta t^2}\right) + \frac{\delta^2}{\delta x^2}(EI(x)\, w_{xx}) = 0 \tag{3}$$

with boundary conditions (2). An equivalent infinite-dimensional model can be generated by representing the flexure w(t, x) with its modes such as

$$w(t,x) = \sum_{i=1}^{\infty} \phi_i(x).q_i(t) \tag{4}$$

where ϕ_i are the mode shapes and q_i are the mode amplitudes. A set of mode shapes can be generated by solving the eigenproblem

$$\frac{d^4 \phi_i}{d x^4} = k_i^4 \phi_i \tag{5}$$

with boundary conditions (2). Let $\theta = (z, q_1, q_2, \ldots)^T$, and assume constant mass density and moment of inertia. Equation (3) can be written in its modal form

$$M \ddot{\theta} + K \theta = F f(t) \tag{6}$$

where

$$M = \begin{bmatrix} 1 & \Phi_1 & \Phi_2 & \ldots \\ \Phi_1 & 1 & & \\ \Phi_2 & & 1 & \\ \vdots & & & \end{bmatrix}, \qquad K = \begin{bmatrix} 0 & 0 & 0 & \ldots \\ 0 & \omega_1^2 & & \\ 0 & & \omega_2^2 & \\ \vdots & & & \end{bmatrix},$$

$$F = \left[\frac{1}{\rho L} \; 0 \; 0 \ldots\right]^T, \quad \omega_i = k_i^2 \left(\frac{EI}{\rho}\right)^{\frac{1}{2}}, \text{ and } \Phi_i = \frac{1}{L}\int_0^L \phi_i(x)\, dx.$$

This in turn can be written as an infinite-dimensional transfer function. Notice that structural damping is not included in this modeling, which results in an infinite number of modes with zero damping in (6). This, however, is not the usual case in practice. For a practical structure there are typically only a finite number of lower frequency modes that need to be controlled. The remaining higher frequency modes will not affect the performance as long as the controller bandwidth is properly limited. We shall assume this is the case, and therefore consider an open-loop plant of finite dimension N.

Suppose a position sensor is placed on the beam at $x = x_j$. Hence, the output equation has the form

$$y = C\,\theta = \left[1 \; \phi_1(x_j) \; \phi_2(x_j) \; \ldots \; \phi_{N-1}(x_j)\right] \theta \tag{7}$$

and the transfer function is

$$\frac{y}{f} = C\,(s^2 M + K)^{-1} F \tag{8}$$

where M, K and F are finite dimensional versions of the corresponding matrices in (6).

It is well known [2] that when the sensor position coincides with the actuator, $x = 0$ in this case, the system's poles and zeros are regularly interlaced on the imaginary axis as shown in Fig. 3. Suppose the sensor is moved from $x = 0$ toward the end $x = L$. Then, the zeros of the system exhibit an interesting pattern of movement [1]. Beginning form the highest mode, the zeros will cross the poles while the poles remain invariant. At some point , the zeros will break into the complex plane in a nonminimum-phase fashion. We note that before the zeros break into complex patterns the system model has the form of Eq. 1 as shown in Fig. 4.

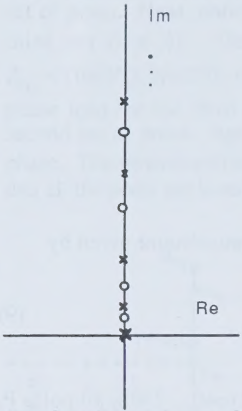


Fig. 3. Plant pole-zero locus for a collocated sensor and actuator.

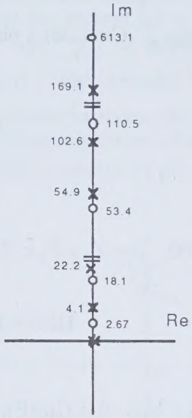


Fig. 4. Plant pole-zero locus of the beam with a sensor located at x = 0.5 in .

III. GENERALIZED LEAD-LAG COMPENSATORS

The main result of this paper is a compensation technique for systems (1). First, we define a notion called k-stability.

Definition 1. Consider an open loop system $G(s) = K\, b(s)/a(s)$ in a unity feedback closed loop system, where $K > 0$, and a(s), b(s) are monic polynomials with $\deg(a) > \deg(b)$. An open-loop pole P, that is $a(P) = 0$, is said to be k-stabilizable, if there exists a non empty interval $(0, K_0)$ such that the root locus originating from P is in the open left half plane for K in $(0, K_0)$. The closed loop system is said to be k-stable if every open-loop pole is k-stabilizable.

K-stability will be the goal that our compensation technique is aimed at. Obviously, a stable closed-loop system is not necessarily k-stable. In what follows we give three lemmas, which are the basis of the proposed compensation. The proofs of the lemmas are based on the classical root locus techniques [4]; they are omitted here and can be found in [1]. The first lemma shows the well-known fact that a lead compensator suffices to stabilize a system (1) in which the interlacing pattern of poles and zeros is not broken.

Lemma 1. Consider a plant

$$G(s) = Kg\,\frac{\prod_{i=1}^{N-1}(s^2+Z_i^2)}{s^2\prod_{i=1}^{N-1}(s^2+P_i^2)}$$

where, $Kg > 0$, and $0 < Z_i < P_i < Z_{i+1}$ for all $i \geq 1$. Then, the unity feedback closed-loop system with a real lead compensator $H(s) = K_h \frac{s+Z}{s+P}$, $0 < Z < P$, $K_h > 0$ is k-stable.

Lemma 2. Consider a plant

$$G(s) = Kg \frac{\prod_{i=1}^{N-1}(s^2+Z_i^2)}{s^2 \prod_{i=1}^{N-1}(s^2+P_i^2)}$$

where, $Kg > 0$, $0 < Z_i < P_i < Z_{i+1}$ for all $i \geq 1$. Let $H(s)$ be a compensator given by

$$H(s) = K_h \frac{(s+Z)(s+Z^*)}{(s+P)(s+P^*)} \tag{9}$$

such that, $K_h > 0$, $0 < (\mathrm{Im}(P), \mathrm{Im}(Z)) < Z_1$, and $\mathrm{Re}(P) > \mathrm{Re}(Z) > 0$. Then, all poles P_i of $G(s)$, $i \geq 1$ are k-stabilizable.

Lemma 3. Consider an open loop system

$$G(s) = Kg \frac{\prod_{i=1}^{N-1}(s^2+Z_i^2)}{s^2 \prod_{i=1}^{N-1}(s^2+P_i^2)}$$

where, $Kg > 0$, $0 < P_i < Z_i < P_{i+1}$ for all $i \geq 1$. Let $H(s)$ be a compensator of the form

$$H(s) = K_h \frac{(s+Z)(s+Z^*)}{(s+P)(s+P^*)} \tag{10}$$

such that, $K_h > 0$, $0 < (\mathrm{Im}(P), \mathrm{Im}(Z)) < P_1$, and $\mathrm{Re}(Z) > \mathrm{Re}(P) > 0$. Then, all poles P_i of $G(s)$, $i \geq 1$ are k-stabilizable.

It is to be noted that the lag compensator cannot stabilize the rigid body mode, i.e. the overall system is not k-stable. We shall call compensator (9) a generalized lead compensator, and compensator (10) a generalized lag compensator.

We now address the problem of stabilizing the class of systems (1). First, we note that to stabilize the rigid body mode, real lead compensation is necessary. Secondly, according to lemmas 2 and 3, each ith subset ($i = 1, ..., r$) of poles can be k-stabilized either by generalized lead or lag compensation (depending on interlacing). These observations suggest the compensator structure depicted in Fig. 5. The real-valued lead pair (P_1^c, Z_1^c) is to stabilize the rigid body mode and the first set of poles ($i = 1$). A generalized lag pair (P_2^c, Z_2^c) is placed so that $P_{n_1} < (\mathrm{Im}(P_2^c), \mathrm{Im}(Z_2^c)) < P_{n_1+1}$, and $\mathrm{Re}(Z_2^c) > \mathrm{Re}(P_2^c) > 0$. Both minimum and nonminimum phase configurations are possible, as shown in Fig. 5 and Fig. 6. This special placement of (P^c_2, Z^c_2) provides the needed phase lag for the second set of poles (i = 2). Furthermore, it results in additional phase lead for the double integrator and the first set of poles. Next, notice the interlacing reversal between the second set (i = 2) and the third set (i = 3). Hence, a generalized lead pair (P^c_3, Z^c_3) is inserted so that $Z_{n_2} < (\mathrm{Im}(P^c_3), \mathrm{Im}(Z^c_3)) < Z_{n_2+1}$, and $\mathrm{Re}(P^c_3) > \mathrm{Re}(Z^c_3) > 0$. This ensures the required phase lead for the third set (i = 3) of poles, and also introduces more phase lag to the second set of poles. Again, the generalized lead can be either minimum or nonminimum phase. The process can be continued in the same manner with a total of r pairs (P^c_i, Z^c_i), so that all the poles are k-stabilized.

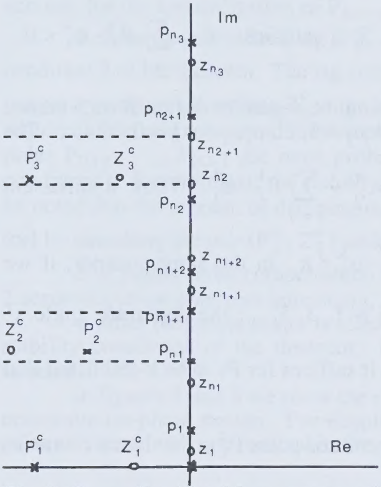

Im
P_{n_3}
Z_{n_3}
P_{n_2+1}
Z_{n_2+1}
P^c_3
Z^c_3
Z_{n2}
P_{n_2}
Z_{n_1+2}
P_{n_1+2}
Z_{n_1+1}
P_{n_1+1}
Z^c_2
P^c_2
P_{n_1}
Z_{n_1}
P_1
P^c_1
Z^c_1
Z_1
Re

Fig. 5. Generalized lead-lag compensation. Minimum-phase design.

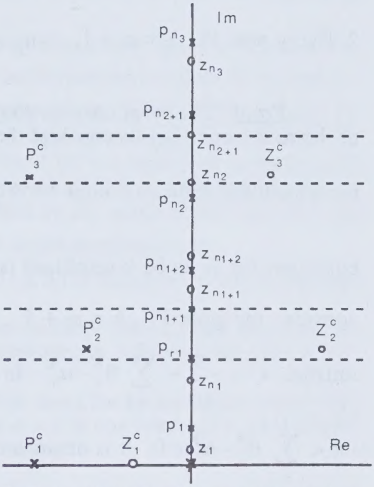

Im
P_{n_3}
Z_{n_3}
P_{n_2+1}
Z_{n_2+1}
P^c_3
Z^c_3
Z_{n2}
P_{n_2}
Z_{n_1+2}
P_{n_1+2}
Z_{n_1+1}
P_{n_1+1}
P^c_2
Z^c_2
P_{n_1}
Z_{n_1}
P_1
P^c_1
Z^c_1
Z_1
Re

Fig. 6. Generalized lead-lag compensation Nonminimum-phase design.

The following theorem gives necessary and sufficient conditions for the compensation procedure to be k-stable.

Let θ^j_i, i = 1, ..., r for j = 1, ..., n_r be the phase between Z^c_i and the plant pole P_j; α^j_i the phase between P^c_i and the pole P_j; x^j the departure angle of the loci branch that starts from P_j. We now state the main theorem.

Theorem. Let G(s) be a system given by (1). Let H(s) be a generalized lead-lag compensator defined by

$$H(s) = \frac{K_h\,(s+Z^c_1)\prod_{i=2}^{r}(s+Z^c_i)\,(s+Z^{c*}_i)}{(s+P^c_1)\prod_{i=2}^{r}(s+P^c_i)\,(s+P^{c*}_i)} \tag{11}$$

such that $K_h > 0$, $P_1^c > Z_1^c > 0$, and

$P_{n_1} < (\mathrm{Im}(P_2^c), \mathrm{Im}(Z_2^c)) < P_{n_1+1}$, and $\mathrm{Re}(Z_2^c) > \mathrm{Re}(P_2^c) > 0$
$Z_{n_2} < (\mathrm{Im}(P_3^c), \mathrm{Im}(Z_3^c)) < Z_{n_2+1}$, and $\mathrm{Re}(P_3^c) > \mathrm{Re}(Z_3^c) > 0$
$P_{n_3} < (\mathrm{Im}(P_4^c), \mathrm{Im}(Z_4^c)) < P_{n_3+1}$, and $\mathrm{Re}(Z_4^c) > \mathrm{Re}(P_4^c) > 0$
$\vdots$

The unity feedback closed loop system is k-stable if and only if

1. Every pole P_k, $k = n_j + 1, \ldots, n_{j+1}$ for $j = 0, 2, 4, \ldots$ satisfies $0 < \sum_{i=1}^{r} \theta_i^k - \alpha_i^k < \pi$.

2. Every pole P_k, $k = n_j + 1, \ldots, n_{j+1}$ for $j = 1, 3, 5, \ldots$ satisfies $-\pi < \sum_{i=1}^{r} \theta_i^k - \alpha_i^k < 0$.

Proof. The proof consists simply of examining the departure angles of each branch of the root locus. Let us consider the pole P_k, $k = n_j + 1, \ldots, n_{j+1}$ for $j = 0, 2, 4, \ldots$ The corresponding departure angle x^k is given by $x^k = \frac{\pi}{2} + \sum_{i=1}^{r} \theta_i^k - \alpha_i^k$. Thus, a sufficient condition for P_k to be k-stabilized is $0 < \sum_{i=1}^{r} \theta_i^k - \alpha_i^k < \pi$. In the same manner, if we consider the pole P_k, $k = n_j + 1, \ldots, n_{j+1}$ for $j = 1, 3, 5, \ldots$; the departure angle x^k satisfies $x^k = -\frac{\pi}{2} + \sum_{i=1}^{r} \theta_i^k - \alpha_i^k$. In other words, it suffices for P_k to be k-stabilized that $-\pi < \sum_{i=1}^{r} \theta_i^k - \alpha_i^k < 0$. It is obvious that the compensator poles P_i^c, $i = 1, \ldots, r$ are k-stabilizable, since they belong to the open LHP.

Now, we need to prove the k-stability of the rigid body mode. For this, we consider the loci branch that departs from the rigid body mode. Let P_e be the closed loop pole that corresponds to a sufficiently small displacement from the point (0, 0) in the upper half of the complex plane. Let α be the phase between P_e and (0, 0), and $(\theta_1^e - \alpha_1^e) > 0$ the phase contribution from the real lead pair (P_1^c, Z_1^c) at the pole P_e. Let the angular contribution from any two symmetric zeros be ξ_i, where ξ_i are small positive numbers. Let the phase contribution from any two symmetric poles be $-\varepsilon_i$, where ε_i are small positive numbers. Thus, the angle α is given by $\alpha = \frac{\pi}{2} + \frac{1}{2}((\theta_1^e - \alpha_1^e) + \sum_i (\xi_i - \varepsilon_i))$. For a small displacement the term $\sum_i (\xi_i - \varepsilon_i)$ is small, therefore it is possible to place the lead (P_1^c, Z_1^c) so that $(\theta_1^e - \alpha_1^e) >> \sum_i (\xi_i - \varepsilon_i)$, i.e. the rigid body mode is k-stabilizable.

It is also clear that conditions 1 and 2 of the theorem are necessary. Since, any violation of conditions (1) and (2) of the theorem implies a loci branch departs in the open RHP. In this case the closed-loop system is not k-stable.

IV. DESIGN PROCEDURE

In this section we describe a procedure for selecting generalized lead-lag compensators for systems (1). A simple example is given to illustrate the design and to show the robust stability.

Design procedure. 1. Start with some real-valued lead (P_1^c, Z_1^c). The zero Z_1^c should be close enough to the origin so that the rigid body mode can be k-stabilized. Some additional phase lead will be introduced by (P_i^c, Z_i^c) $i \geq 2$, and should be taken into account for the k-stabilization of $P_1, ..., P_{n1}$.

2. Select a generalized lag (P_2^c, Z_2^c) to stabilize P_{n1+1} and P_{n2}, i.e. by satisfying condition 2 of the theorem. The lag contribution can be increased by pulling P_2^c toward the imaginary axis and stretching Z_2^c to the left of the complex plane. If (P_i^c, Z_i^c) $i \geq 3$ will be required, then allow some additional phase lag. When P_{n1+1} and P_{n2} are k-stabilized, the poles $P_{n1+2}, ..., P_{n2-1}$ are most probably k-stabilized (it is guaranteed under certain conditions). Nevertheless, the stability of every mode has to be checked (step 4). It is to be noted that the amount of damping may be controlled by the compensator gain K_h, and (or) by stretching the pair (P_2^c, Z_2^c) which results in a larger departure angle.

3. If higher order compensation (i.e. (P_i^c, Z_i^c) $i \geq 3$) is necessary , then repeat step 2 according to the pole-zero interlacing.

4. After the compensator is selected, one has to be check that all poles satisfy the k-stability conditions of the theorem. Finally, an appropriate compensator gain K_h is selected.

In figures 7 and 8 we show the shape of the root locus for the minimum-phase and nonminimum-phase design. For simplicity, the case $r = 2$ is considered, i.e. two sets of poles and zeros with one reversal in the pole-zero interlacing.

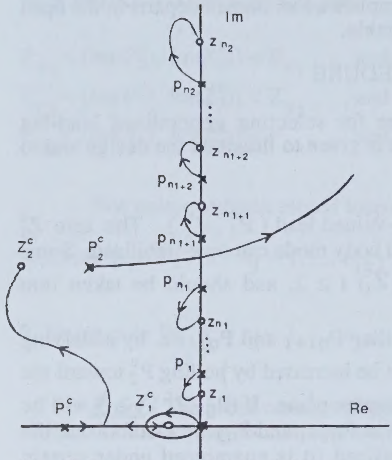


Fig. 7. Root locus of the minimum-phase design.

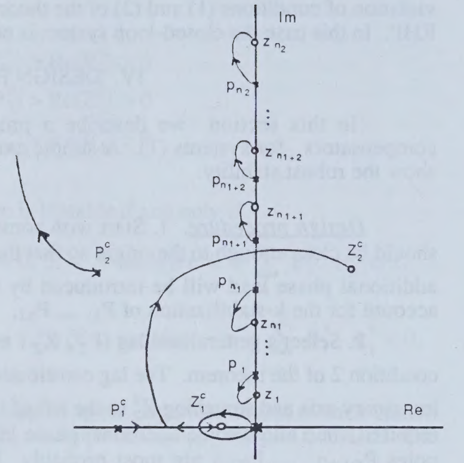


Fig. 8. Root locus of the nonminimum-phase design.

Design example. To illustrate the proposed compensation technique we consider the control of the flexible beam of figure 2, which has a transfer function as shown in figure 4. Following the procedure, a generalized lead-lag controller is found to be $H = K_h \dfrac{(s + 0.2)(s - 250 + j\,100)(s - 250 - j\,100)}{(s + 100)(s + 250 + j\,100)(s + 250 - j\,100)}$. Simulation results for $K_h = 100000$ are presented in figures 9 - 13 and table 1. They show a satisfactory performance with adequate robustness.

V. CONCLUSION

In this paper we have presented a design methodology for a specific class of dynamical systems. The control synthesis is relatively simple and can provide a low-order and robust compensation of the high order plant. The method handles poles and zeros on the imaginary axis, which are in a pattern not easy to compensate for. An important application is flexible beam control when exact collocation cannot be achieved. There is a considerable amount of flexibility in the choice of the pole-zero topology of the compensator. Typically, several trials are required to obtain satisfactory damping for each mode.

REFERNCES

[1] Arbouz, N.M., " Robust control of flexible structures", Ph.D. dissertation, University of Colorado at Boulder (in preparation).

[2] Martin, G.D., " On the control of flexible mechanical systems", Ph.D. dissertation, Guidance and control laboratory, Stanford University, 1978.

[3] Kimura, H., "Robust stabilizability for a class of transfer functions", IEEE Trans. on Automatic control, Vol. AC-29, No. 9, pp. 788-793, September 1984.

[4] Franklin, G.F.; Powel, J.D.; Emami-Naeini, A., Feedback control of dynamic systems, Addison-Wesley, 1986.

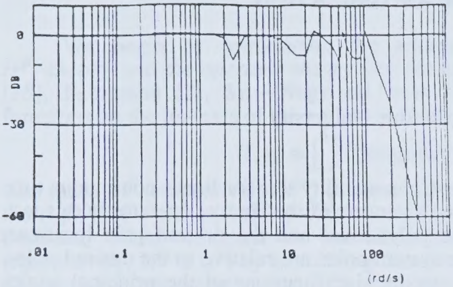


Fig. 9. Magnitude frequency response of the closed-loop system.

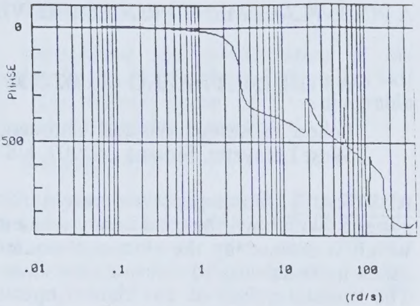


Fig. 10. Phase response of the closed-loop system.

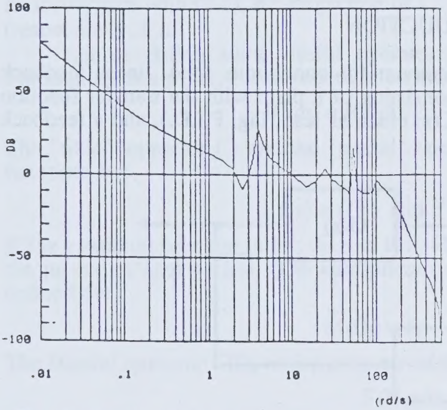


Fig. 11. Magnitude frequency response of the open-loop system GH(s).

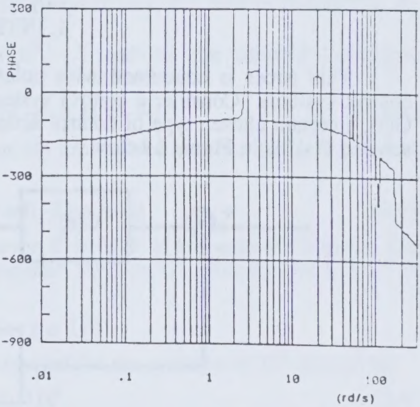


Fig 12. Phase response of the open-loop system GH(s).

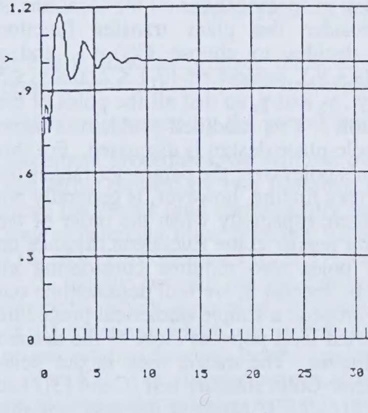


Fig 13. Step-response of the closed-loop system.

Closed loop poles of compensated system	
No parameter variations	25% drop in all frequency modes ω_i
- 0.209	- 0.24 ± j 0.2
- 0.47 ± j 2.69	- 0.32 ± j 3.1
- 4.1 ± j 21.6	- 0.28 ± j 16.8
- 30.1 ± j 5.9	- 0.03 ± j 41.2
- 0.33 ± j 55.6	- 0.02 ± j 76.8
- 1.1 ± j 101.3	- 95.8
- 3.4 ± j 169.5	- 0.34 ± j 126.6
- 260.5 ± j 106.9	- 250.8 ± j 100.5

Table 1. Effect of model parameter variations upon stability of closed-loop system.

A POLE-PLACEMENT PROCEDURE VIA H^∞-OPTIMIZATION

ARTHUR E. FRAZHO and DZU K. LE

School of Aeronautics and Astronautics, Purdue University
West Lafayette, Indiana , 47907, USA

ABSTRACT. In this paper we present a Schur-Cohn stability test for linear continuous time systems, which will be used in pole-placement. The norm of the Hankel operator in this test, which is defined by the system characteristic polynomial and the desired pole locations, offers a natural way to measure how close the system poles are relative to the desired poles. The singular values of this Hankel operator are the sine functions of the principal angles between the H^2-subspaces generated by the two set of poles. A stability test of the Jury-Routh type based on the Schur numbers is also presented for the purpose of pole-placement. The pole-placement procedures, derived here from these stability tests, appear to be particularly helpful for systems with high M^{c}Millan degree.

1. INTRODUCTION

This paper is concerned with pole-placement for continuous time, linear feedback control systems. Consider a control system consisting of a plant with the transfer function G(s), a certain chosen type of control action C(s) (such as lead, lag, P.I.D.), and a feedback scheme F(s), as in Figure 1 below.

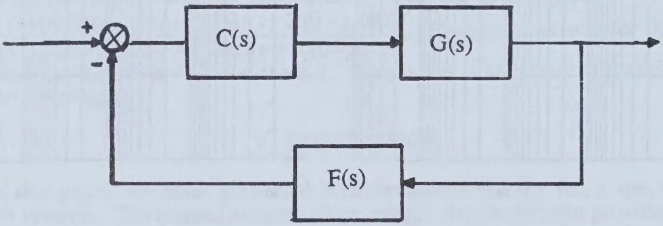


Figure 1. The Standard Feedback Control System.

Within the various constraints imposed on the selectability of the parameters of such a control scheme, it is often very difficult to satisfy the design specifications on the locations of the closed-loop poles. For example, let us consider the plant transfer function $G(s) = (s+1)^2/s(s+3)$. Suppose that the engineer decides to choose $C(s) \equiv 1$, and a feedback compensation of the form $F(s) = \gamma_1(s+\gamma_2)/(s+\gamma_3)$, subject to $|\gamma_1| \leq 2$, $0 \leq \gamma_2 \leq 5$ and $1.5 \leq \gamma_3 \leq 2.5$. How would he or she then choose γ_1, γ_2 and γ_3 so that all the poles of the closed-loop system are as close to, say, -2.0 as possible? This modified problem is taken from Hostetter and Santina [11], where the algebraic pole-place design is discussed. For this simple example one may resort to extracting the system's poles over the parameter ranges for an extensive search of a solution. Relying on explicit root finding, however, is generally not appropriate or not even possible for this kind of problem; especially when the order of the system is too high. Moreover, explicit assessment, with regard to the Euclidean distance on the s-plane, of the closeness between two sets of poles also requires considering all permutations of the poles, which is time-consuming. In Section 8, we will demonstrate our pole-placement procedures on this example. Here, we propose a simple numerical procedure for selecting parameter values in order to place the closed-loop poles as close to the desired pole locations as possible with respect to a natural metric. The metric used in our pole-placement procedure is inspired by the well known Schur-Cohn stability test (Cohn [3]) and the related works of Young [21], and Pta'k and Young [16], [17]. Many of our results in this paper rest on the H^∞-optimization theorem of Nehari [13] and its generalization by A.A.K. [1].

2. PRELIMINARY

We begin by introducing the necessary background and nomenclature of *the H^p-theory* and *the operator theory* (see for examples, Duren [5], Hoffman [10], Nikol'skii [15], Fuhrmann [7], Sz. – Nagy and Foias [20]). The Hilbert space of L^2 – integrable functions on the jω–axis of the s-plane is denoted by L^2; with the inner product

$$(f, g) \equiv \int_{-\infty}^{+\infty} f(j\omega)\overline{g(j\omega)}d\omega \qquad (\text{for } f, g \in L^2) \quad .$$

The space H^2 (respectively, H^∞) denotes the usual Hardy space of holomorphic functions on the right-half s-plane (R.H.P.) which are uniformly L^2-bounded (respectively, uniformly L^∞-bounded) when restricted on the lines parallel to the jω–axis. We have $H^2 \subset L^2$ (respectively, $H^\infty \subset L^\infty$) by identifying the elements of this space with their boundary values. The anti–H^2 (respectively, anti–H^∞), that is the Hardy-space similar to H^2 (respectively, H^∞) defined on the left-half s-plane, will be denoted by K^2 (respectively, K^∞). By the Paley - Wiener theorem (see Nikol'skii [15], Hoffman [10]):

$$L^2 = H^2 \oplus K^2 \tag{2.1}$$

In accordance with (2.1), the projection of L^2 on H^2 (respectively, K^2) will be denoted by P_+ (respectively, P_-).

Let us define some useful operators on L^2, L^∞ and on the rational functions. Throughout this paper, J is the operator on either L^2 or L^∞ defined by:

$$Jf(j\omega) \equiv f(-j\omega) \qquad (\text{for } -\infty < \omega < -\infty) \tag{2.2}$$

The "tilda" operator ($\tilde{}$) and the "pound" operator (#) are defined on the family of rational functions by:

$$\tilde{f}(s) \equiv \overline{f(\overline{s})} \;\text{ and }\; f^{\#}(s) \equiv \overline{f(-\overline{s})} = \tilde{f}(-s) \tag{2.3}$$

If f is a rational function in H^2, then so is $\tilde{f}$. However, $f^{\#}$ is in K^2 if and only if f is in H^2. On the jω–axis $f^{\#}(j\omega) = \overline{f(j\omega)}$. The *multiplication operator* M(f) *on* L^2, *with symbol* f *in* L^∞ is defined by:

$$M(f)g \equiv fg \qquad (\text{for } g \in L^2)$$

The *Hankel operator* Γ(f), *with a proper rational symbol* f is the operator on H^2 defined by:

$$\Gamma(f) \equiv P_+M(f)J \mid H^2 \tag{2.4}$$

where the vertical "bar" means "restriction to".

Let us recall some further definitions and notations. A *Blaschke product* b *of length* n is a rational function of the form:

$$b(s) = \prod_{i=1}^{n} (s - \alpha_i)/(s + \overline{\alpha_i}) \quad (\text{where } \mathrm{Re}\alpha_i > 0 \quad \text{for all } i) \tag{2.5}$$

If b is a Blaschke product, the space H(b) always means $H^2 \ominus bH^2$; that is, the orthogonal complement of bH^2 in H^2. It is well known that H(b) is an n-dimensional subspace of H^2, which are spanned by the partial fractions of $1/\prod_{i=1}^{n}(s + \overline{\alpha_i})$, see Nikol'skii [15]. In our pole-placement procedures, we will use the principal angles between the two H^2-subspaces $H(b_1)$ and $H(b_2)$ defined by two Blaschke products b_1 and b_2, as a natural way to measure how close the zeroes (and consequently the poles) of b_1 and b_2 are to one another. However, the way to calculate these principal angles (defined in Section 4) is via the singular values of $\Gamma(b_1\, b_2^{\#})$. This operator $\Gamma(b_1\, b_2^{\#})$ is also motivated by the Schur-Cohn test of Theorem 3.1.

Finally, we will use the following version of the H^∞–optimization theorem (Nehari's theorem), which appears as "Problem $A^{[k]}$" in A.A.K. [1]. To this end let K_j^∞ denote the set of all functions which are representable in the form of a sum r(s) + h(s), where r(s) is a proper rational function having j poles (counting multiplicity) which are all in the L.H.P. and h is in K^∞. The singular values of an operator Γ will be denoted by σ_k, and are listed in decreasing order, starting with $\sigma_o = \|\Gamma\|$. This sets the stage for the following result:

THEOREM (A.A.K. [1]). *Let* f *be a proper rational function, with no poles on the* jω-axis. *Then*

$$\min_{g \in K_j^\infty} \|f - g\|_\infty = \sigma_j$$

where σ_j *is the* j^{th}*-singular value of* $\Gamma(f)$.

3. A HANKEL OPERATOR VERSION OF THE SCHUR-COHN TEST.

In this section we will present a Hankel operator version of the Schur-Cohn test ([19], [3], [17]). This will motivate our pole-placement procedure. To this end recall that a polynomial p(s) is said to be *strict Hurwitz* if all of its zeroes are in the open L.H.P. In the following theorem, for any polynomial p(s), the notation b_p stands for the rational function

$$b_p(s) \equiv p(-s)/\bar{p}(s) \ . \tag{3.1}$$

THEOREM 3.1. *Let* p(s) *be a polynomial of degree* n *and* b(s) *be any Blaschke product of length* n. *Then* p(s) *is strict Hurwitz if and only if*

$$\|\Gamma(b\, b_p^{\#})\| < 1 \ .$$

PROOF. First, assume that $\|\Gamma(b\, b_p^{\#})\| < 1$. By Nehari's theorem this assumption implies that:

$$\min_{g \in K^\infty} \|b\, b_p^{\#} - g\|_\infty < 1 \tag{3.2}$$

Consequently, by multiplying the expression $(b\, b_p^{\#} - g)$ by $b^{\#}$ and noting that $b^{\#} b = 1$, there exists a function h in H^∞ such that

$$\|b_p - bh\|_\infty < 1 \ . \tag{3.3}$$

Consider a Nyquist contour D which contains all the zeroes in the R.H.P. of b and b_p. By taking D large enough we may also assume that $|b_p|$ is very close to one on the part of D that is off the jω–axis because $|b_p(\infty)| = 1$. Moreover, the modulus of b_p is equal to one on the jω–axis. Thus, due to inequality (3.3), by a proper choice of D we have:

$$|b_p(s) - b(s)h(s)| < |b_p(s)| \quad (\text{for all } s \in D) \tag{3.4}$$

By the *argument principle* [18] inequality (3.4) implies that:

$$nz(b_p) - np(b_p) = nz(b) + nz(h) \ ; \tag{3.5}$$

where "nz" (respectively, "np") denotes the *number of zeroes* (respectively, *number of poles*) within D, of the corresponding functions. Using $nz(b) = n$ and $nz(b_p) \le n$, it follows from (3.5) that $np(b_p) = 0 = nz(h)$, and $nz(b_p) = n$. This means that all the zeroes of p are in the L.H.P; see the definition (3.1).

Conversely, assume that p(s) is a strict Hurwitz polynomial. Since $\|\Gamma(b\, b_p^{\#})\| \le 1$ it remains to show that $\|\Gamma(b\, b_p^{\#)}\|$ cannot equal to one. Let us proceed by contradiction. If $\|\Gamma(b\, b_p^{\#})\| = 1$, then there exists two vectors x and y in H^2 such that

$$y = \Gamma(b\, b_p^{\#})x \tag{3.6}$$

Thus, we have

$$1 = \|y\| = \|P_+(b\, b_p^{\#} Jx)\| \le \|b\, b_p^{\#} Jx\| = \|x\| = 1 \ .$$

Consequently,

$$\|y\| = \|P_+(b\, b_p^{\#} J x)\| \ . \tag{3.7}$$

Since P_+ is an orthogonal projection equality (3.7) implies that

$$y = b\, b_p^{\#} Jx \ . \tag{3.8}$$

It is easy to verify that the range of $\Gamma(b\, b_p^{\#})$ is orthogonal to bH^2; thus, the range is contained in H(b). Since H(b) is generated by n rational functions, equation (3.6) implies that y is rational; and so is x, by equation (3.8). Now, as a rational function in H^2, all the poles of x have to be in the open L.H.P. Moreover, the relative degree of x must be less than one. Consequently, in the R.H.P. we must have:

$$\mathrm{nz}(Jx) < \mathrm{np}(Jx) \ . \tag{3.9}$$

On the other hand, since y is in H^2, expression (3.8) shows that in the R.H.P. the zeroes of b have to cancel all the poles of $b_p^{\#}Jx$. But, this is not possible because by inequality (3.9) we have in the R.H.P.:

$$\mathrm{np}(b_p^{\#}\,Jx) \geq \mathrm{np}(b_p^{\#}) + \mathrm{np}(Jx) - \mathrm{nz}(Jx) > \mathrm{np}(b_p^{\#})$$

that is, $\mathrm{np}(b_p^{\#}Jx) > n$, which means there are more poles in $b_p^{\#}Jx$ than b could possibly cancel. This contradiction shows that $\|\Gamma(b\, b_p^{\#})\|$ is strictly less than one, which completes the proof.

4. BLASCHKE PRODUCTS - NATURAL METRIC; PRINCIPAL ANGLES.

Let us introduce the following function ρ on the family of pairs of Blaschke products,

$$\rho(b_1, b_2) \equiv \|\Gamma(b_1\, b_2^{\#})\| \qquad (\text{if length } b_1 = \text{length } b_2)$$

$$\equiv 1 \qquad\qquad (\text{otherwise}). \tag{4.1}$$

It will be shown that if P_1 and P_2 denote the projections of H^2 onto $H(b_1)$ and $H(b_2)$, respectively, then

$$\rho(b_1, b_2) = \|P_1 - P_2\| \tag{4.2}$$

Consequently, since each Blaschke product b is uniquely determined by its associated space H(b) (Theorem 4.1 (d) below), expression (4.2) shows that ρ is *naturally a metric* on the family of all Blaschke products. This is essentially due to Pta'k and Young [16]. Classically, $\|P_1 - P_2\|$ is known as the aperture of Range P_1 and Range P_2 (Akhiezer and Glazman [2]). It turns out that other singular values of $\Gamma(b_1\, b_2^{\#})$ are also related to the principal angles of $H(b_1)$ and $H(b_2)$. We extend the definition of principal angles in Golub and VanLoan [9] to the case of two finite dimensional subspaces M_1 and M_2 of a Hilbert space as follows.

DEFINITION 4.1. Let M_1 and M_2 be two non-identical finite dimensional subspaces of a Hilbert space H, and assume that $n \equiv \dim M_1 \leq \dim M_2$. Then the *non-zero principal angles* $\theta_1, \theta_2, \ldots, \theta_{n-d}$ (where $d \equiv \dim M_1 \cap M_2$; and where $\theta_k \in [0, \frac{\pi}{2}]$) of M_1 and M_2 and their *principal directions* $u_k \in M_1$, $v_k \in M_2$ are defined inductively by:

$$\cos\theta_k = \max|(u, v)| = (u_k, v_k) \quad ;$$
$$u \in M_1 \ominus (M_1 \cap M_2)$$
$$v \in M_2 \ominus (M_1 \cap M_2)$$

subject to $\|u\| = \|v\| = 1$, and $(u, u_i) = 0 = (v, v_i)$ for $i = 1, 2, 3, \ldots, k-1$.

Our main theorem of this section relies on the following lemma, in which P_1 and P_2 denote *the orthogonal projection of* H^2 *onto* $H(b_1)$ *and* $H(b_2)$, respectively; and the *identity operator* on H^2 is denoted by I.

LEMMA 4.1. *Given two Blaschke products* b_1 *and* b_2, *not necessarily of the same length. Then, the non-zero singular values of the following operators are the same:*

(a) *The Hankel operator* $\Gamma(b_1\, b_2^{\#})$;

(b) *The operator* $P_1M(b_2)\,|\,H(b_1)$; *and*

(c) *The operator* $(I - P_2)P_1$.

SKETCH OF PROOF. From the commutativity $JP_+ = P_-J$ and the fact that $P_1 = M(b_1)P_-M(b_1^{\#})\,|\,H^2$ (consult Nikol'skii [15]; also $(I - P_1) = M(b_1)P_+M(b_1^{\#})\,|\,H^2$), one can easily deduce that:

$$M(b_1)J\Gamma(\tilde{b}_1\tilde{b}^{\#}) = P_1M(b_2)\,|\,H^2 \ . \tag{4.3}$$

Observe that $\Gamma(\tilde{b}_1\tilde{b}_2^{\#})$ is the adjoint of $\Gamma(b_1\, b_2^{\#})$. Moreover, $M(b_1)J$ is unitary; and, $P_1M(b_2)\,|\,H^2$ vanishes on b_1H^2. Hence, expression (4.3) gives the first two results of Lemma

4.1. On the other hand $JM(b_1^{\#})$ unitarily maps $H(b_1)$ onto $H(\tilde{b}_1)$, see Fuhrmann [7]. Using this, and taking the adjoint of both sides of (4.3) we have on $H(b_1)$:

$$\Gamma(b_1 b_2^{\#}) JM(b_1^{\#}) \mid H(b_1) = M(b_2^{\#})(I - P_2) \mid H(b_1) \ . \tag{4.4}$$

Again, the relationship (4.4) implies that Lemma 4.1 is also true for the two operators in (a) and (c). The proof is now complete.

THEOREM 4.1. *Let* b_1 *and* b_2 *be two Blaschke products. Assuming* $n_1 \equiv$ length $b_1 \le n_2 \equiv$ length b_2, *then*

(a) *The sine function of the non-zero principal angles of* $H(b_1)$ *and* $H(b_2)$ *are the non-zero singular values of* $\Gamma(b_1 b_2^{\#})$.

(b) *The leading* $(n_2 - n_1)$ *singular values of* $\Gamma(b_2 b_1^{\#})$ *are equal to* 1, *while the remaining of its non-zero singular values are the same as those of* $\Gamma(b_1 b_2^{\#})$.

(c) $\|\Gamma(b_2 b_1^{\#})\| = \|P_1 - P_2\|$.

(d) *If the singular value* σ_k *of* $\Gamma(b_1 b_2^{\#})$ *is zero, and if* σ_{k-1} *is non-zero then* b_1 *and* b_2 *share exactly* $n_1 - k$ *common zeroes.*

SKETCH OF PROOF. Essentially, from the development in Golub and Vanloan [9] we can show that the sine function of the non-zero principal angles of $H(b_1)$ and $H(b_2)$ are the non-zero singular values of the operator $(I - P_2)P_1$ in Lemma 4.1(c). Hence, Theorem 4.1(a) is proved. Part (b) and (c) are also the result of this development and of Lemma 4.1(c). Part (d) follows from the fact that, if b_1 and b_2 have no common zero then Range $\Gamma(b_1 b_2^{\#}) = H(b_1)$ (Douglas, Shapiro and Shield [4]). Thus, the proof of Theorem 4.1 is completed.

The claim (4.2) follows immediately from Theorem 4.1 (b) and (c).

5. PRINCIPAL ANGLES AND ROOT-CONTINUITY

The following continuity theorems confirm that the singular values of $\Gamma(b f^{\#})$ are valid criteria for pole-placement:

THEOREM 5.1. *Let* b *be a fixed Blaschke product. Let* f *denote an arbitrary Blaschke product, the length of which is not necessarily the same as that of* b. *If* $\|\Gamma(b f^{\#})\|$ *approaches zero, then each zero of* b *is approached by at least a zero of* f, *counting multiplicity.*

REMARK 5.1. Due to space limitation, we omit the proofs of this theorem and of Theorem 5.2 below as well. However, these theorems are just simple applications of Nehari's theorem, the Maximum Modulus Principle, and Rouche's theorem (Rudin [18]).

THEOREM 5.2 *Let* b *be a fixed Blaschke product of length* n. *Let* f *denote an arbitrary Blaschke product with a length at least equals* n. *If the* k^{th} *singular value* σ_k *of* $\Gamma(b f^{\#})$ *approaches zero (where* $0 \le k < n$), *then the zero-set of* f *approaches that of* b *along at least* $n-k$ *pairs of elements from each set.*

6. A MATRIX REPRESENTATION FOR $\Gamma(b f^{\#})$

In this section we will present a fast algorithm for computing the matrix representation of the Hankel operator of the form $\Gamma(b f^{\#})$ in certain unitary bases; where b is a Blaschke product, and f is a proper, stable rational function. Since Range $\Gamma(b f^{\#}) \subseteq H(b)$ and Null $\Gamma(b f^{\#}) \supseteq \tilde{b}H^2$, which is the orthogonal complement in H^2 of $H(\tilde{b})$, we only need to find a matrix representation for the restriction:

$$\Gamma(b f^{\#}) \mid H(\tilde{b}) : H(\tilde{b}) \to H(b) \tag{6.1}$$

Let us introduce the following notations assuming that $b(s) = \prod_{i=1}^{n} (s - \alpha_i)/(s + \bar{\alpha}_i)$.

$$d_i \equiv (\mathrm{Re}\,\alpha_i/\pi)^{1/2} \quad \text{for all } i \tag{6.2}$$

$$B_i(s) \equiv (s - \alpha_i)/(s + \bar{\alpha}_i) \qquad \text{(for all i)} \tag{6.3}$$

For $1 \le i \le j \le n$, let

$$\phi_{i,j}(s) = (d_i/(s + \bar{\alpha}_i)) \prod_{i<k\le j} B_k(s) \tag{6.4}$$

$$\psi_{i,j}(s) = (d_j/(s + \bar{\alpha}_j)) \prod_{i\le k<j} B_k(s) \tag{6.5}$$

Then, for each j and each $i \le j$, the collections $\{\phi_{k,j}\}_{k=i}^{k=j}$ and $\{\psi_{i,k}\}_{k=i}^{k=j}$ are both orthonormal bases of $H(B_i\, B_{i+1} \cdots B_j)$ (see Nikol'skii [15]). We can also verify this by *the reproducing property of the kernel* $1/2\pi(s + \bar{\alpha}))$ *for* H^2*–functions ; i.e.,*

$$(h, 1/2\pi(s + \bar{\alpha})) = h(\alpha) \text{ for all } h \in H^2 \text{ and } \operatorname{Re} \alpha > 0 \tag{6.6}$$

From property (6.6), we can also show that, if we use $\{\tilde{\phi}_{k,n}\}_{k=1}^{k=n}$ as an *orthonormal basis for* $H(\tilde{b})$ and $\{\psi_{1,k}\}_{k=1}^{k=n}$ as an *orthonormal basis for* $H(b)$, then the *matrix representation of* $\Gamma(b\,f^{\#})$ on $H(b)$ is an *upper-triangular matrix* $[\gamma_{i,j}]$ such that:

$$\bar{\gamma}_{i,j} = -\,(f\phi_{j,j}, \phi_{i,j}) \tag{6.7}$$

The expression (6.7) can be evaluated very quickly by the following algorithm (letting $g = f\,\phi_{j,j}$)

THEOREM 6.1 : *Let* g *be an element of* H^2, *then for* $1 \le i \le j \le n$,

$$(g, \phi_{i,j}) = 2\pi d_i g_{i,j}(\alpha_i) \tag{6.8}$$

where the $g_{k,j}$ *(for* $k=j, j-1, \ldots, i$) *are the* H^2–functions *which are defined recursively as:*

$$g_{j,j}(s) \equiv g(s) \text{ and } g_{k-1,j}(s) = [g_{k,j}(s)\,(s + \bar{\alpha}_k) - 2\pi d_k^2 g_{k,j}(\alpha_k)]/(s - \alpha_k) \tag{6.9}$$

REMARK 6.1: The point $s = \alpha_k$ is a removable singular point of the right-hand-side of expression (6.9); see (6.2). If g is rational then $g_{k-1,j}$ is simply given by the Euclidean algorithm from (6.9).

SKETCH OF PROOF: From $H^2 = H(B_k) \oplus B_k H^2$, and from the reproducing property (6.6), we can write any H^2–function, say h, as (given any $1 \le k \le n$):

$$h = 2\pi d_k h(\alpha_k)\phi_{k,k} + B_k h_k \text{ , where } h_k \in H^2 \text{ .} \tag{6.10}$$

In particular, given that $g_{k,j}$ is known, there exists a unique $g_{k-1,j}$ in H^2 such that:

$$g_{k,j} = 2\pi d_k g_{k,j}(\alpha_k)\phi_{k,k} + B_k g_{k-1,j} \tag{6.11}$$

which agrees with formula (6.9). Thus, applying expression (6.11) repeatedly, starting from k=j, we have:

$$g(s) = 2\pi d_j g_{j,j}(\alpha_j)\phi_{j,j}(s) + 2\pi d_{j-1} g_{j-1,j}(\alpha_{j-1})\phi_{j-1,j}(s) +$$

$$\cdots + 2\pi d_i g_{i,j}(\alpha_i)\phi_{i,j}(s) + B_i(s)B_{i+1}(s) \cdots B_j(s) g_{i-1,j}(s) \tag{6.12}$$

Expression (6.12) implies immediately (6.8), and the theorem is proved.

REMARK 6.2. In our applications the zeroes of b, namely the α_k's, are known. If g is any proper rational function then g admits a factorization of the form $g = b\,f^{\#}$ where b is a Blaschke product and f is in H^∞. However, for a general Hankel operator $\Gamma(g)$ if the zeroes of b are not accurately given, then one has to rely on a more complicated algorithm to get a matrix representation of $\Gamma(g)$; such as the algorithm given in Glover [8].

7. A JURY-ROUTH TEST WITH SCHUR NUMBERS

We present in this section a stability test of the Jury-Routh type which is very simple and numerically reliable. This test also gives rise to a pole-placement procedure. To this end, let us give a generalization of the Schur numbers. Let p(s) be a polynomial of degree n. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be an arbitrary sequence of numbers with $\operatorname{Re} \alpha_i > 0$ for all i. We define *the Schur numbers of* p(s) *associated with the data* $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ by:

$$\tau_i \equiv p_i(-\alpha_{i+1})/\tilde{p}_i(\alpha_{i+1}) \quad (\text{for } i=0, 2, \ldots, n-1) \tag{7.1}$$

where the polynomials p_i'*s are defined recursively as:*

$$p_o(s) \equiv p(s)\ , \text{ and } p_{i+1}(s) \equiv (p_i(-s) - \tau_i \tilde{p}_i(s))/(s - \alpha_{i+1}) \tag{7.2}$$

Note that by expression (7.1), the point $s = \alpha_{i+1}$ is a removable singular point of p_{i+1} as defined in (7.2). It turns out that we have this *stability test,* which is easy to prove:

$p(s)$ *is strict Hurwitz if and only if all the Schur numbers* τ_i (for i=0, 1, ..., n−1) have modulus strictly less than one.

REMARK 7.1. Let $f_i(s) \equiv p_i(-s)/\tilde{p}_i(s)$. Then, the f_i's are related to one another via Moebius transformations and Blaschke products if $|\tau_i| < 1$ for i=0, 1, ..., n−1. Namely,

$$f_{i+1}(s) = \frac{f_i(s) - \tau_i}{1 - \bar{\tau}_i f_i(s)} \; \frac{s + \bar{\alpha}_{i+1}}{s - \alpha_{i+1}} \tag{7.3}$$

Furthermore, f_i is, modulo a constant of modulus one, a Blaschke product of length n-i. Now, let

$$E_k = \prod_{i=0}^{n-k-1} (1 - |\tau_i|^2) \quad (\text{for } k=0, 1, 2, \cdots, n-1) \tag{7.4}$$

Then, *if* E_o *approaches* 1, the *zeroes of* p(s) *approach the "desired" locations* $\{-\alpha_1, -\alpha_2, ..., -\alpha_n\}$ in the similar manner as described by Theorem 5.1. Likewise, if E_k approaches 1, then n-k zeroes of p(s) approach the "desired" locations in the similar manner as described by Theorem 5.2. This continuity phenomenon suggests that the E_k's may be used in place of the Euclidean metric for pole-placement. However, E_k depends on the ordering of the desired pole locations. These numbers are related to the notion of the *"Entropy of functions"*. Otherwise, E_k does not have a geometric interpretation such as the singular values of $\Gamma(b f^{\#})$.

8. APPLICATION TO POLE-PLACE DESIGN

From the development of the earlier sections, we see that the singular values of $\Gamma(b f^{\#})$ (see (2.4)) and the numbers $E_k(p; \alpha_1, \alpha_2, ..., \alpha_n)$ (see (7.4)) are useful criteria for assessing the closeness between certain set of desired pole-locations, $\{-\alpha_1, -\alpha_2, \cdots, -\alpha_n\}$, and that of the system poles in the L.H.P. wherein,

$$b(s) \equiv \prod_{i=1}^{n} (s - \alpha_i)/(s + \bar{\alpha}_i) \tag{8.1}$$

$$f^{\#}(s) \equiv p(s)/p(-s) \tag{8.2}$$

and p(s) is the closed-loop system characteristic polynomial of degree n. Here, we emphasize that all the coefficients of p are real, and that p(s) carries parameters which are to be selected by the engineer for system stability and desireable closed-loop poles profile. The point of our pole-placement procedure is to minimize one of the singular values σ_k (for k=0, 1, ..., n−1); or, to maximize one of the numbers E_k (for k=0, 1, $\cdots$, n−1). By Theorem 5.2, and similar observations in Section 7, if σ_k is sufficiently small or if E_k is sufficiently close to 1, then n-k of the system poles are close to n-k of the $-\alpha_i$'s. For the Hankel method, we can also use a reduced set of desired poles, say $\{-\alpha_1, -\alpha_2, .., -\alpha_k\}$ $(k < n)$, and the resulting

$$b(s) \equiv \prod_{i=1}^{k} (s - \alpha_i)/(s + \bar{\alpha}_i) \tag{8.3}$$

instead of the Blaschke product b of full length n as in (8.1). In this case the aim is to minimize this $\|\Gamma(b f^{\#})\|$. If the size of the reduced set of poles is not too large, the accuracy and the computing time for $\|\Gamma(b f^{\#})\|$ is not affected noticeably by the order of the system; for, the dimension of Range $\Gamma(b f^{\#})$ is less than or equal to the length of b, and our algorithm (Theorem 6.1) involves only a few polynomial divisions. For a system of high M^{c}Millan degree, and a choice of b with a great length, k, it is advisable to use E_{n-k}. We emphasize here, again, that E_k depends on the ordering of the set $\{\alpha_1, \cdots, \alpha_n\}$. We demonstrate our pole-place procedure by a few examples below. Here we use the stability criteria in Section 7 to screen out parameters values that render instability to the system. The simplex method for function minimization (see Nelder and Mead [14]) was employed with our criterion $\|\Gamma(b f^{\#})\|$, or E_o, on the Gould computer, using a few random initial guesses, to get the following results.

EXAMPLE 1: Let $G(s) = (s+3)/(s+1)(s+2)$; $C(s) \equiv \gamma$ $(0 < \gamma < 10)$ and $F(s) \equiv 1$. Referring to Figure 1., $p(s) = (s+1)(s+2) + \gamma(s+3)$. If we take the desired poles as $-1.5 \pm 2j$, then the Hankel norm is minimized at $\gamma = 1.417$. This corresponds to $\|\Gamma(b f^{\#})\| = 0.19101$, which places the system poles at $-2.2083 \pm 1.1719j$. This is a very acceptable solution because, by plotting out the root locus of p(s) one sees that these poles are almost the orthogonal projection of the desired poles on the root-locus. That is, these poles are practically the closest one can get to the desired poles.

EXAMPLE 2. Let $G(s) = (s^2 + 2s + 4)/s\ (s+4)\ (s+6)\ (s^2 + 1.4s + 1)$; $C(s) \equiv \gamma$ $(0 \leq \gamma \leq 200)$; $F(s) \equiv 1$. Here, we have a fifth-order system. If we only want to have two poles close to $-1 + 3j$, and two other poles to $-1 - 3j$ then the Hankel method yields $\gamma = 119.3427$, with $\|\Gamma(b f^{\#})\|$ equal to 0.62008 (here length $b = 4$), which places the poles at $-.146 \pm 3.064j$, $-1.17 \pm 2.101j$, and -8.762. The first two poles appear to be a little far from $-1 \pm 3j$. However, due to the root locus of p(s) of this case, with a small branch very close to the $j\omega$–axis, this is the best one can get even by the Euclidean distance observation. The transfer function G(s) of this example is taken from Ogata [22] (page 345). The E_1 criterion yields $\gamma = 122.2213$, with the corresponding $\|\Gamma(b f^{\#})\|$ (at this value γ) equal to 0.6232. This places the poles at $-.141 \pm 3.12j$, and $-1.16 \pm 2.09j$. Hence, the two answers are compatible. The rest of the examples also shows the agreement of the two criteria $\|\Gamma(b f^{\#})\|$ and E_o $(p, \alpha_1, \alpha_2, \ldots, \alpha_k)$, where $\alpha_1, \cdots \alpha_k$ are the zeroes of b.

EXAMPLE 3. We now consider the example in the introduction. Here the system characteristic polynomial is:

$$p(s) = s(s+3)(s+\gamma_3) + \gamma_1(s+\gamma_2)(s+1)^2 \text{ ; where}$$

$$|\gamma_1| \leq 2 \text{ ; } 0 \leq \gamma_2 \leq 5 \text{ and } 1.5 \leq \gamma_3 \leq 2.5 \text{ .}$$

(a) If we take the desired poles to be -2.0 with multiplicity 3, then the Hankel method yield $\gamma_1 = -1.6111$, $\gamma_2 = 3.1724$, $\gamma_3 = 1.5$. This corresponds to $\|\Gamma(b f^{\#})\| = .022$, which places the poles at $-1.354 \pm .716j$ and -3.5643. This is quite acceptable.

(b) If we just want two poles close to -2.0, then the solution is: $\gamma_1 = +1.070245$, $\gamma_2 = 1.65249$, $\gamma_3 = 1.81404$. This places two poles at -2.0 and the third at $-.2135$.

(c) If no restriction is made on γ_1, γ_2, γ_3, then by the Hankel method, the values $\gamma_1 = -1.57143$, $\gamma_2 = 2.9091$, $\gamma_3 = 1.2857$ will put all the poles very close to -2.0.

EXAMPLE 4. Here, G(s) is the same as in Example 3, but $C(s) \equiv 1$ and $F(s) = \gamma$, with $|\gamma| \leq 10$. Then

(a) with the desired poles at -2.0 and -1.0, the Hankel solution is $\gamma = -6.016$; which places the poles at $-.9 \pm .6j$

(b) if only one pole is required to be close to -2, the Hankel solution is $\gamma = 2$; which places the poles at -2.0 and $-.333$.

9. CONCLUSION

It was demonstrated in this paper that, whenever a pole-place design is feasible for certain control scheme, the singular values of the Hankel operator $\Gamma(b f^{\#})$, and the Schur numbers of the closed-loop system – regarding the set of desired poles reflected by b – are good alternatives to the Euclidean distance. A clear advantage of these pole-placement criteria is that, they do not require explicit pole-extraction. Although the background and the motivation of these criteria are extensive, their computing-algorithms (Section 6 and 7) are quite explicit, involving only polynomial long-divisions and a singular-value decomposition. The proposed pole-placement procedures, therefore, can be easily automated, incorporating a simple stability check (such as in Section 7) and a numerical optimization scheme (such as in Nelder and Mead [14]). A few words of caution is due here. If at the optimal solution the criterion is too close to the stability limit (i.e.; 1 for $\sigma_k(\Gamma(b f^{\#}))$, and zero for the E_k numbers), then one should consider reducing the number of desired pole locations or using the higher singular values.

REFERENCES

[1] V.M. Adamjan, D.Z. Arov and M.G. Krein, *Analytic Properties of Schmidt Pairs for a Hankle Operator and the Generalized Schur-Takagi Problem,* Math. USSR Sbornik, Vol. 15 (1971), No. 1.

[2] N.I. Akhiezer and I.M. Glazman, *Theory of Linear Operators in Hilbert Space,* Monographs and Studies in Mathematics, Pitman Advanced Program.

[3] A. Cohn, Über *die Anzahn der Wurzeln einer algebraischen Gleichung in einem Kreise,* Math Z., Vol. 14, pp. 110-148, (1922).

[4] R.G. Douglas, H.S. Shapiro, and A.L. Shields, *Cyclic Vectors and Invariant Subspaces for the Backward Shift,* Ann. Inst. Fourier, Grenoble 21, 1, pp. 37-76, (1971).

[5] P.L. Duren, *Theory of* H^p *spaces,* New York - London: Academic Press, (1970).

[6] B.A. Francis, *A Course in* H_∞ *Control Theory,* Lecture Notes in Control and Information Sciences, Vol. 88, Springer-Verlag, New York, (1987).

[7] P.A. Fuhrmann, *Linear Systems and Operator in Hilbert Space,* Advanced Book Program, McGraw-Hill, (1981).

[8] K. Glover, *All Optimal Hankel Norm Approximations of Linear Multivariable Systems and Their* L^∞*-error Bounds,* Int. J. Control, Vol. 39 (1984), No. 6, 1115-1193.

[9] G.H. Golub and C.F. VanLoan, *Matrix Computations,* The Johns Hopkins University Press, Second Printing, (1984).

[10] K. Hoffman, *Banach Spaces of Analytic Functions,* Englewood Cliffs, N.J., Prentice Hall, (1962).

[11] G.H. Hostetter and M.S. Santina, *Rational Linear Algebraic Tracking Control System Design,* IEEE Cont. Syst. Magazine, Vol. 8, N°4, pp. 34-42, (Aug. 1988).

[12] E.I. Jury, *Inners and Stability of Dynamic Systems,* New York: Wiley, (1974).

[13] Z. Nehari, *On Bounded Bilinear Forms,* Ann. of Math. (2), 65 (1957), pp. 153-162.

[14] J.A. Nelder and R. Mead, *A Simplex Method for Function Minimization,* Computer J.7, pp. 308-313 (1965).

[15] N.K. Nikol'skii, *Treatise on the Shift Operator, Spectral Function Theory,* A Series of Comprehensive Studies in Math. Vol. 273, Springer-Verlag, New York. (Translated from the Russian by Jaak Peetre) (1986).

[16] V. Pta'k and N.J. Young, *Function of Operators and the Spectral Radius,* Linear Algebra and its Applications, Vol. 29 (1980), 357-392.

[17] V. Pta'k and N.J. Young, *A Generalization of the Zero Location Theorem of Schur and Cohn,* IEEE Transaction on Automatic Control, Vol. AC-25, No. 5, (October 1980).

[18] W. Rudin, *Real and Complex Analysis,* McGraw-Hill, New York, (1964).

[19] I. Schur, *Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind,* J. für Math., Vol. 147, pp. 205-232, (1917), and Vol. 148, pp. 122-145, (1918).

[20] B. Sz.-Nagy and C. Foias, *Harmonic Analysis of Operators on Hilbert Space,* Budapest, Hungary: Akademiai Kiado , (1970).

[21] N.J. Young, *Norm of Certain Rational Functions of a Matrix and Schur's Criterion for Polynomials,* Comment. Math. Univ. Carolinae, Vol. 19, pp. 673-687, (1978).

[22] K. Ogata, *Modern Control Engineering,* Engineering Series, Prentice-Hall, Inc., Englewood Cliffs, N.J., (1970).

ROBUSTNESS OF FEEDBACK SYSTEMS WITH SEVERAL SOURCES OF MODELLING ERROR

J.S. FREUDENBERG* and J. CHEN*
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122

Abstract

In this short paper, we outline an approach to robustness analysis for feedback systems with several sources of uncertainty.

The study of feedback system robustness has several important goals. First, one needs to *analyze* whether the stability and performance properties of a system are robust against modelling errors. Second, one needs to be able to *design* so that robustness is present. Third, there is a need to determine inherent *limitations* upon robustness properties, and tradeoffs between robustness and performance. Obtaining generally applicable statements about such problems is probably a hopeless task. Yet it seems reasonable to expect that one could define special *classes* of design problems with a few (e.g. 2, 3, 4) sources of uncertainty, and make general statements about these. For the general problem with 2 sources of uncertainty, this procedure was carried out in [1], and applied to the specific problem of analyzing performance robustness when the plant is ill-conditioned. One result was a set of guidelines useful in design and in assessing inherent design limitations.

In this short paper, we outline a procedure for extending the results of [1] to problems with more than two sources of uncertainty. Our approach, as in [1], is to use Doyle's structured singular value (SSV) to characterize robustness. Recall that this technique requires the block diagram of the feedback system to be rearranged into the form of a block-diagonal uncertainty matrix placed in a feedback loop around an interconnection matrix which depends upon the plant, compensator, and various weighting matrices. It is instructive to consider the following (academic) question. Under what conditions can a system with n sources of uncertainty have arbitrarily *large* stability margins against any combination of k uncertainties, $k < n$, yet have arbitrarily *small* stability margin against all n uncertainties? This question can be resolved by an appeal to Mason's Gain Rule to analyze the flow of signals in the block diagram used for SSV analysis. The key step is to divide the loops through this diagram according to their length, which is equal to the number of uncertainties each loop contains. Making the system robust to arbitrarily large combinations of k uncertainties corresponds to making sure that each loop of length k has zero gain. Furthermore, we can characterize those interconnection matrices that can only be destabilized by a combination of all n uncertainties. Of course, when applied to a specific design problem, or class of design problems, the elements of the interconnection matrix are *not* independent. Hence for a given problem, one can analyze whether insuring robustness against, say, all combinations of one or two uncertainties also guarantees robustness against all combinations of three uncertainties.

---

*Supported by AFOSR Contract No. F33615-88-C-3601.

The procedure just outlined yields interesting results when applied to a robust performance problem which was discussed by Skogestad and Morari. In this modification, it is noted that physical considerations often rule out uncertain coupling between different actuators. Hence the unstructured uncertainty used to represent actuator modelling errors is constrained to be diagonal or block diagonal. With the uncertainty block used to represent performance, this naturally results in a problem with at least three sources of uncertainty. Our analysis shows that, for this special class of problems, requiring robustness against uncertainties taken one or two at a time does tend to insure that the system is robust against all three uncertainties.

[1] J.S. Freudenberg, "Analysis and design for ill-conditioned plants - Part I and II" to appear, *International Journal of Control*, 1988.

REPRESENTATION OF UNCERTAIN ACTUATOR DYNAMICS AND DECENTRALIZED ROBUST CONTROLLER DESIGN

ALTUĞ İFTAR
Department of Electrical Engineering, University of Toronto
Toronto, Canada M5S 1A4

ÜMİT ÖZGÜNER
Department of Electrical Engineering, The Ohio State University
Columbus, Ohio 43210

Abstract: Representation of uncertain actuator dynamics is discussed within a recently introduced framework. It is shown that, under mild conditions on uncertain actuator dynamics, it is possible to obtain a rational transfer function matrix (TFM), possibly dependent on a parameter vector which varies over a subset of a finite dimensional real vector space, to represent such dynamics. Once a TFM is obtained, a state space realization of it can be combined with the system model and a controller can be designed to guarantee stability and good performance. Here the system model itself may also be subject to parameter variations. Extension of this approach to decentralized control systems and decentralized robust controller design are also discussed. It is demonstrated that local controllers which guarantee overall stability and desired performance can be designed, if certain limited information about rest of the closed-loop system is available to each local agent.

I. INTRODUCTION

One of the fundamental issues in feedback controller design is robustness. Since an exact model of a physical system would be very complicated if not impossible to obtain, the designer should base the controller design on a *nominal* model and should require the controller to perform satisfactorily under possible deviations from the nominal model. At this point modeling of such deviations (*i.e.*, *uncertainties*) becomes a crucial issue. The uncertainties in a mathematical model of a physical system are basically due to two factors: unknown values of certain system parameters (*e.g.*, resistance of an electrical component) and totally unmodeled dynamics (*e.g.*, high frequency modes of a flexible structure). It has been generally accepted that state space models are more suitable for representing uncertainties due to parameter variations, while unmodeled dynamics can be represented easier in the frequency domain.

Recently, much research has been undertaken to establish frequency domain design strategies for systems with uncertain dynamics [1]–[7], on the one hand, and to develop state space design methods for systems with parameter uncertainties [8]–[18], on the other hand. In [19] Wei and Yedavalli proposed a combined approach for systems which exhibit uncertainties of both classes. Recently, a unified approach in state space was presented by the authors [20].

In the present paper, representation of uncertain actuator dynamics and decentralized robust controller design are discussed within the framework of [20]. A background on modeling of uncertain dynamics for robust controller design is presented in Section II. Representation of uncertain actuator dynamics and robust controller design is discussed in Section III. Modelling of actuator dynamics for decentralized systems is considered in Section IV. Decentralized robust controller design is discussedd in Section V. Some concluding remarks are included in Section VI.

In the sequel $\mathbf{C}$ denotes the space of complex numbers, $\mathbf{R}$ denotes the space of real numbers, $\mathbf{R}^k$ denotes the k dimensional real vector space, and I denotes the identity matrix of appropriate dimensions. Furthermore, $\overline{(\cdot)}$, $|\cdot|$, and $\angle(\cdot)$ denote, respectively, the complex conjugate, magnitude, and phase of $(\cdot)$.

II. REDUCED ORDER MODELS FOR UNCERTAIN DYNAMICS

The uncertain dynamics of a system, nominally modeled by $G(s)$, can be represented at the input by $\Delta_i(s)$, at the output by $\Delta_o(s)$, additively by $\Delta_a(s)$, or in feedback form by $\Delta_f(s)$, as shown in Figure 1. Hereafter, we will genericly denote any such representation by $\Delta(s)$.

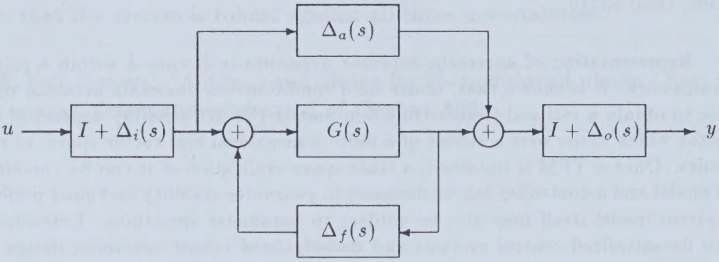


Figure 1: Representation of uncertain dynamics

Although $\Delta(s)$ may not be known exactly, some information about it (*e.g.*, upper and lower bounds on $\Delta(j\omega)$) is generally available. Furthermore, $\Delta(s)$ may not be representable as a finite dimensional linear system, or such a representation may require an unmanageably high dimensional model.

In [20] it was shown that, under mild conditions on uncertain dynamics, it is possible to obtain a rational transfer function matrix (TFM) $E(s;p)$ to replace $\Delta(s)$ in the model, such that a feedback controller $K(s)$ designed for such a model stabilizes the actual plant and achieves desired performance. Here $E(s;p)$ is possibly parametrized by a vector p which is taken from a parameter set Π. The parameter set Π can be chosen as a subset of a finite dimensional real vector space. These results are summarized, for the case of uncertainties represented at the output ($\Delta_o(s)$ in Figure 1), by the following theorems. The results can easily be extended to other types of representations. The proofs can be found in [20].

Theorem 1a (*Absolute Stability*): Suppose that $\Delta(s)$ is analytic in the closed right half complex plane $\mathbf{C}^+$, except possibly at some isolated singular points which constitute a total of m poles with due count of multiplicity.[1] Let $E(s;p)$ be a TFM which is rational in s and continuous in p and Π be a subset of a finite dimensional real vector space with the properties:

a) for all $\omega \in \mathbf{R}$ there exists a $p_\omega \in \Pi$ such that

$$E(j\omega;p_\omega)=\Delta(j\omega) \tag{1a}$$

if $\Delta(s)$ is analytic at $s=j\omega$ or

$$\lim_{r\to 0} E(j\omega+re^{j\theta};p_\omega)=\lim_{r\to 0}\Delta(j\omega+re^{j\theta}) \tag{1b}$$

for all $-\frac{\pi}{2}\le\theta\le\frac{\pi}{2}$ if $\Delta(s)$ is not analytic at $s=j\omega$,

---

[1] Note that, one needs to know the total number of unstable poles of $\Delta(s)$. Their actual locations need not be known.

b) there exists a $p_\infty \in \Pi$ such that

$$\lim_{r \to \infty} E(re^{j\theta}; p_\infty) = \lim_{r \to \infty} \Delta(re^{j\theta}) \tag{1c}$$

for all $\frac{\pi}{2} \geq \theta \geq -\frac{\pi}{2}$, and

c) $E(s;p)$ has exactly m poles in $\mathbf{C}^+$ for all $p \in \Pi$.

For a given rational TFM $G(s)$, suppose that a rational TFM $K(s)$ is chosen such that the closed–loop TFM (CLTFM)

$$T_E(s;p) \triangleq (I + E(s;p))G(s)K(s)[I + (I + E(s;p))G(s)K(s)]^{-1} \tag{2}$$

does not have any poles in $\mathbf{C}^+$ for all $p \in \Pi$. Also assume that $\det[I + (I + \Delta(s))G(s)K(s)] \neq 0$ on a dense subset of $\mathbf{C}$. Then the true CLTFM

$$T_\Delta(s) \triangleq (I + \Delta(s))G(s)K(s)[I + (I + \Delta(s))G(s)K(s)]^{-1} \tag{3}$$

is analytic in $\mathbf{C}^+$.

Note that $T_\Delta(s)$ represents a physical system whenever $\Delta(s)$ does. Furthermore, $T_\Delta(s)$ represents the actual closed–loop system whenever $\Delta(s)$ is the TFM representing the actual unmodeled dynamics. Thus, whenever $T_\Delta(s)$ is analytic in $\mathbf{C}^+$, the actual closed–loop system is stable.

In many practical cases, absolute stability may not be sufficient. Instead, for example, one may wish to confine all the closed–loop eigenvalues in a region $\mathcal{R}$ of the complex plane. Note that the imaginery axis and the right hand semi–circle of infinite radius together defines the boundary of $\mathbf{C}^+$. Hence, if the conditions a) and b) of Theorem 1a are met, we say that $E(s;p)$ and $\Delta(s)$ are *matched* on the boundary of $\mathbf{C}^+$. Furthermore, we say that the two TFMs are matched on the boundary of a region $\mathcal{R} \subset \mathbf{C}$, if the obvious generalizations of these conditions hold for the region $\mathcal{R}$.

Theorem 1b (*Relative Stability*): Let $\mathcal{R}$ be an open subset of $\mathbf{C}$. Suppose that $\Delta(s)$ is analytic in $\mathcal{R}^c \triangleq \mathbf{C} \setminus \mathcal{R}$, except possibly at some isolated singular points which constitute a total of m poles with due count of multiplicity. Let $E(s;p)$ be a TFM which is rational in s and continuous in p and Π be a subset of a finite dimensional real vector space such that $E(s;p)$ and $\Delta(s)$ are matched on the boundary of $\mathcal{R}^c$ and $E(s;p)$ has exactly m poles in $\mathcal{R}^c$. For a given rational TFM $G(s)$, suppose that a rational TFM $K(s)$ is chosen such that the closed–loop TFM (2) does not have any poles in $\mathcal{R}^c$ for all $p \in \Pi$. Also assume that $\det[I + (I + \Delta(s))G(s)K(s)] \neq 0$ on a dense subset of $\mathbf{C}$. Then the true closed–loop TFM (3) is analytic in $\mathcal{R}^c$.

It is well known that most of the widely used performance measures (*e.g.*, disturbance rejection and steady state error) can be related to the return difference matrix [2]. For example, to achieve a certain degree of plant disturbance rejection at the output, one may require the return difference matrix to satisfy[2]:

$$[I + (I + \Delta(j\omega))G(j\omega)K(j\omega)]^S \geq Q(\omega) \qquad \forall \omega \in \Omega \tag{4}$$

provided that the left hand side is well defined. Here $Q(\omega)$ is a positive definite matrix for all $\omega \in \Omega$, $\Omega \subset \mathbf{R}$ is the set of frequencies where disturbances are effective, for appropriately dimensioned matrices A and B, $A^S \triangleq A^H A$, A^H denotes the complex conjugate transpose of A, and $A \geq B$ means $A - B$ is positive semi-definite. The following theorem demonstrates that if a controller is designed to satisfy such a performance criterion for the *design model* (the model in which Δ is replaced by E), then the actual closed-loop system satisfies the same criterion.

---

[2] The more widely used criterion is defined as a lower bound on the singular values of the return difference matrix. However, note that (4) is a more general condition.

Theorem 1c (*Good Performance*): Under the conditions of Theorem 1a, suppose $K(s)$ is chosen such that

$$[I + (I + E(j\omega;p))G(j\omega)K(j\omega)]^S \geq Q(\omega) \qquad \forall\omega \in \Omega\ , \quad \forall p \in \Pi\ . \tag{5}$$

Then (4) is satisfied.

Under mild conditions on $\Delta(s)$, the existence of a TFM $E(s;p)$ and a parameter set Π, satisfying the conditions of the above theorems, can be ensured. The following theorem presents the desired result for absolute stability in the scalar case. Extensions to relative stability and multivariable case are possible along similar lines.

Theorem 2: Let $\Delta(s) = \frac{1}{d_u(s)}\Delta_s(s)$, where $d_u(s)$ is an m^{th} order polynomial with zeros in $\mathbf{C}^+$ and $\Delta_s(s)$ is analytic in $\mathbf{C}^+$ (*i.e.*, factor out the unstable poles of $\Delta(s)$ as $\frac{1}{d_u(s)}$). Furthermore, suppose that $\Delta(s)$ is such that $\Delta(j\omega) = \overline{\Delta(-j\omega)}$ for all $\omega \in \mathbf{R}$, $\lim_{r\to\infty} \Delta(re^{j\theta}) = c < \infty$[3] for all $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$, and

$$-k\frac{\pi}{2} \leq \angle(\Delta_s(j\omega)) + n\pi \leq \ell\frac{\pi}{2} \qquad \forall\omega \in \mathbf{R}^+ \tag{6}$$

where the equalities can hold only at $\omega = 0$ or as $\omega \to \infty$. Here k and ℓ are non-negative integers, n is an integer such that $\angle(\Delta_s(0)) + n\pi = 0$, and $\mathbf{R}^+$ is the set of non-negative real numbers. Let $q \triangleq \max(k + m, \ell)$ and

$$t \triangleq \begin{cases} \ell & \text{if } c \text{ is known to be zero} \\ q & \text{otherwise} \end{cases} .$$

Then there exists a function $E(s;p)$, continuous in p and proper and rational in s with q poles and t finite zeros, and a set $\Pi \subset \mathbf{R}^{q+t+1}$ such that the conditions a), b), and c) of Theorem 1a hold.

Once a rational TFM $E(s;p)$ is obtained to replace $\Delta(s)$, a state space representation of it can be obtained. Such a representation can be combined with the nominal model, which may also be subject to parameter variations. Hence, what is obtained is a larger dimensional state space model which is subject to parameter variations (see [20] for details). A controller can be designed by considering such a model and the stability of the resulting closed-loop system can be tested by one of the many existing methods. Alternatively, a special controller design method (*e.g.*, [11], [16]-[18]) can be adopted to design a controller which guarantees stability and desired performance.

III. REPRESENTATION OF UNCERTAIN ACTUATOR DYNAMICS

When a controller is to be designed for a physical system, the dynamics of the actuators and sensors to be used must also be considered. A controller design which ignores such dynamics may be disastrous in many cases. Actuator or (usually to a lesser extent) sensor dynamics can destabilize an otherwise stable system (*e.g.*, see [21]). In this section we consider representation of uncertain actuator dynamics for robust controller design. A similar approach can be undertaken for sensor dynamics as well.

---

[3] The actual value of the constant c need not be known.

The uncertain dynamics of an actuator can be represented by $\Delta(s) \triangleq \left[\begin{array}{cc} \Delta_u(s) & \Delta_v(s) \end{array} \right]$, as shown in Figure 2 (if certain parts of the actuator can be modeled accurately such a model can be augmented with the system model). Here u is the control signal, f is the output of the actuator which is applied to the system, y is the system output, and v is a certain collection of the system states which reacts against the actuator (*e.g.*, as back–emf or friction). The existence of such a feedback from the system to the actuator is very common and neglecting this effect may easily cause unstability.

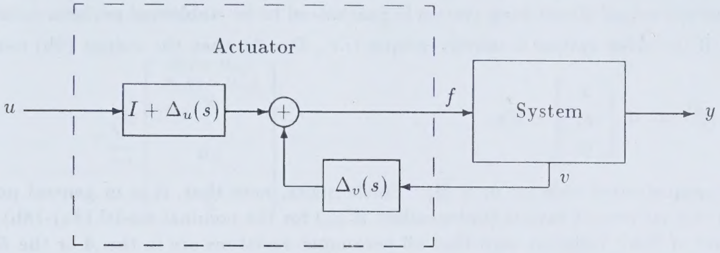


Figure 2: Representation of uncertain actuator dynamics

Although $\Delta(s)$ may not be known exactly, some information about it can be obtained and a rational TFM $E(s; p_a)$ can be found to replace it as discussed in the previous section. Here, p_a is a parameter vector which varies over a parameter set Π_a.

Let $(A_a(p_a), [B_a\ G_a](p_a), C_a(p_a), [D_a\ F_a](p_a))$ be a realization for $E(s; p_a)$, *i.e.*, the actuator is modeled as:

$$\dot{x}_a = A_a x_a + B_a u + G_a v \tag{7a}$$

$$f = C_a x_a + (I + D_a)u + F_a v\ . \tag{7b}$$

Suppose that a nominal model for the system is given by

$$\dot{x} = Ax + Bf \tag{8a}$$

$$y = Cx + Df \tag{8b}$$

$$v = Hx \tag{8c}$$

where the system matrices A, B, C, D, and H may be parameterized by $p_1 \in \Pi_1$ reflecting possible parameter variations in the system. Furthermore, suppose that the uncertain dynamics of the system are represented at the input and a rational TFM $E_s(s; p_2)$ is found to replace such dynamics. Here p_2 is a parameter vector which varies over a set Π_2. Let $(A_s(p_2), B_s(p_2), C_s(p_2), D_s(p_2))$ be a realization for $E_s(p_2)$. Then the overall system dynamics, including the actuator is represented by:

$$\frac{d}{dt}\begin{bmatrix} x \\ x_s \\ x_a \end{bmatrix} = \begin{bmatrix} A + \bar{B}F_aH & BC_s & \bar{B}C_a \\ B_sF_aH & A_s & B_sC_a \\ G_aH & 0 & A_a \end{bmatrix}\begin{bmatrix} x \\ x_s \\ x_a \end{bmatrix} + \begin{bmatrix} \bar{B}(I + D_a) \\ B_s(I + D_a) \\ B_a \end{bmatrix} u \tag{9a}$$

$$y = [C + \bar{D}F_aH \quad DC_s \quad \bar{D}C_a] \begin{bmatrix} x \\ x_s \\ x_a \end{bmatrix} + \bar{D}(I + D_a)u \tag{9b}$$

where $\bar{B} \triangleq B(I + D_s)$ and $\bar{D} = D(I + D_s)$. The system matrices of the above model are parameterized by $p \triangleq (p_1, p_2, p_a)$. A controller can now be designed for this model guaranteeing closed–loop stability and desired performance for all parameter variations over $\Pi \triangleq \Pi_1 \times \Pi_2 \times \Pi_a$. Then, resulting actual closed–loop system is guaranteed to be stable and perform satisfactorily. Note that if the given system is strictly–proper (*i.e.*, $D = 0$), then the output (9b) reduces to

$$y = [C \quad 0 \quad 0] \begin{bmatrix} x \\ x_s \\ x_a \end{bmatrix} = Cx$$

which is parameterized only by $p_1 \in \Pi_1$. Furthermore, note that, it is in general possible to choose a constant output matrix (independent of p_1) for the nominal model (8a)–(8b) by using a proper set of state variables such that all parameter variatons are in the A or the B matrix. In this case the output matrix of the overall system model would be constant as well which may simplify the design process.

IV. MODELING DECENTRALIZED SYSTEMS

The modeling approach outlined above can be extended to decentralized systems. Suppose it is desired to find decentralized controllers for a system nominally modeled by:

$$\dot{x} = Ax + \sum_{i=1}^{N} B_i f_i \tag{10a}$$

$$y_i = C_i x + \sum_{j=1}^{N} D_{ij} f_j \ , \qquad i = 1, 2, \ldots, N \tag{10b}$$

$$v_i = H_i x \ , \qquad i = 1, 2, \ldots, N \tag{10c}$$

where the system matrices may be subject to parameter variations represented by $p_1 \in \Pi_1$. Suppose the uncertain dynamics are represented at the input by $\Delta_s(s)$, and a rational TFM $E_s(s; p_2)$ is found to replace $\Delta(s)$. Let $E_s(s; p_2)$ be realized as:

$$\dot{x}_s = A_s x_s + \sum_{i=1}^{N} B_{s_i} f_i \tag{11a}$$

$$w_i = C_{s_i} x_s + \sum_{j=1}^{N} D_{s_{ij}} f_j \tag{11b}$$

where all the matrices are parameterized by $p_2 \in \Pi_2$.

In this case there exist N independent actuators, each of which can be modeled similar to (7a)–(7b):

$$\dot{x}_{a_i} = A_{a_i} x_{a_i} + B_{a_i} u_i + G_{a_i} v_i \tag{12a}$$

$$f_i = C_{a_i} x_{a_i} + (I + D_{a_i}) u_i + F_{a_i} v_i \tag{12b}$$

where all the matrices are parameterized by $p_{a_i} \in \Pi_{a_i}$ $(i = 1, 2, \ldots, N)$. Now the overall system model can be obtained as:

$$\frac{d}{dt}\begin{bmatrix} x \\ x_s \\ x_{a_1} \\ \vdots \\ x_{a_i} \\ \vdots \\ x_{a_N} \end{bmatrix} = \begin{bmatrix} A + \sum_{i=1}^{N} \bar{B}_i F_{a_i} H_i & \sum_{i=1}^{N} B_i C_{s_i} & \bar{B}_1 C_{a_1} & \cdots & \bar{B}_i C_{a_i} & \cdots & \bar{B}_N C_{a_N} \\ \sum_{i=1}^{N} B_{s_i} F_{a_i} H_i & A_s & B_{s_1} C_{a_1} & \cdots & B_{s_i} C_{a_i} & \cdots & B_{s_N} C_{a_N} \\ G_{a_1} H_1 & 0 & A_{a_1} & & & & \\ \vdots & \vdots & & \ddots & & 0 & \\ G_{a_i} H_i & 0 & & & A_{a_i} & & \\ \vdots & \vdots & & 0 & & \ddots & \\ G_{a_N} H_N & 0 & & & & & A_{a_N} \end{bmatrix} \begin{bmatrix} x \\ x_s \\ x_{a_1} \\ \vdots \\ x_{a_i} \\ \vdots \\ x_{a_N} \end{bmatrix}$$

$$+ \sum_{i=1}^{N} \begin{bmatrix} \bar{B}(I + D_{a_i}) \\ B_{s_i}(I + D_{a_i}) \\ 0 \\ \vdots \\ B_{a_i} \\ \vdots \\ 0 \end{bmatrix} u_i$$

$$y_i = \begin{bmatrix} C_i + \sum_{j=1}^{N} \bar{D}_{ij} F_{a_j} H_j & \sum_{j=1}^{N} D_{ij} C_{s_j} & \bar{D}_{i_1} C_{a_1} & \cdots & \bar{D}_{ij} C_{a_j} & \cdots & \bar{D}_{i_N} C_{a_N} \end{bmatrix} \begin{bmatrix} x \\ x_s \\ x_{a_1} \\ \vdots \\ x_{a_j} \\ \vdots \\ x_{a_N} \end{bmatrix}$$

$$+ \sum_{j=1}^{N} \bar{D}_{ij}(I + D_{a_j}) u_j \ , \quad i = 1, \ldots, N$$

where $\bar{B}_i \overset{\triangle}{=} B_i(I + \sum_{j=1}^{N} D_{s_{ij}})$, $\bar{D}_{ij} \overset{\triangle}{=} D_{ij}(I + \sum_{j=1}^{N} D_{s_{ij}})$. Here the system matrices are parameterized by $p \overset{\triangle}{=} (p_1, p_2, p_{a_1}, \ldots, p_{a_N})$, which varies over $\Pi \overset{\triangle}{=} \Pi_1 \times \Pi_2 \times \Pi_{a_1} \times \cdots \times \Pi_{a_N}$. Once again, if the given decentralized system is strictly–proper, the output equations reduce to:

$$y_i = C_i x \ , \qquad i = 1, \ldots, N$$

and are independent of any parameter variations if the nominal model has constant output matrices.

The model thus generated may be used to design decentralized controllers using one of the available techniques. One technique that can be utilized, which resorts to the same philosophy of state space realization of unknown dynamics, is outlined in the next section.

V. DECENTRALIZED ROBUST CONTROLLER DESIGN

In this section we introduce a multimodelling approach for decentralized robust controller design. To avoid notational complexity, here we consider a large scale system which is composed of only two subsystems and dynamic interconnections. After applying a certain expansion (see [22] for details), such a system can be described by:

$$\dot{\hat{x}}_i = \hat{A}_i \hat{x}_i + \hat{A}_{ij} \hat{x}_j + \hat{B}_i u_i \tag{13a}$$

$$y_i = \hat{C}_i \hat{x}_i \ , \qquad i, j = 1, 2 \ , \quad j \neq i \ . \tag{13b}$$

Suppose a controller, described by:

$$\dot{z}_2 = F_2 z_2 + G_2 y_2 \tag{14a}$$

$$u_2 = H_2 z_2 + K_2 y_2 \, , \tag{14b}$$

is designed by the second control agent. Then the overall system dynamics can be described by:

$$\frac{d}{dt}\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \hat{A}_1 & \hat{A}_{12} & 0 \\ \hat{A}_{21} & \hat{A}_2 + \hat{B}_2 K_2 \hat{C}_2 & \hat{B}_2 H_2 \\ 0 & G_2 \hat{C}_2 & F_2 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ z_2 \end{bmatrix} + \begin{bmatrix} \hat{B}_1 \\ 0 \\ 0 \end{bmatrix} u_1 \tag{15a}$$

$$y_1 = \hat{C}_1 \hat{x}_1 \, . \tag{15b}$$

Note that the controller or even the subplant dynamics corresponding to the second control agent may not be known to the first agent. However a certain amount of information, such as upper and lower bounds of the TFMs corresponding to these dynamics, may be available. By defining

$$\Delta(s) = \begin{bmatrix} \hat{A}_{12} & 0 \end{bmatrix} \left(sI - \begin{bmatrix} \hat{A}_2 + \hat{B}_2 K_2 \hat{C}_2 & \hat{B}_2 H_2 \\ G_2 \hat{C}_2 & F_2 \end{bmatrix} \right)^{-1} \begin{bmatrix} \hat{A}_{21} \\ 0 \end{bmatrix} , \tag{16}$$

a design model, as depicted in Figure 3, is obtained for the first control agent.

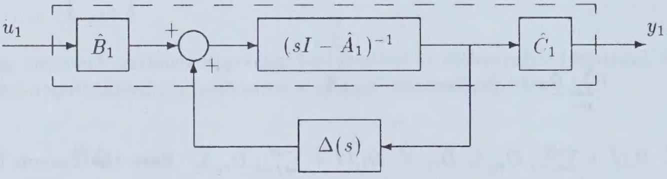


Figure 3: Design model for the first control agent

By knowing certain bounds on $\Delta(s)$, a rational TFM $E(s;p)$ parametrized by a vector $p \in \Pi$ and a parameter set Π can be determined to represent $\Delta(s)$, as discussed in Section II. Let $(F(p), G(p), H(p), L(p))$ be a realization for $E(s;p)$. Then

$$\frac{d}{dt}\begin{bmatrix} x \\ \xi \end{bmatrix} = \begin{bmatrix} \hat{A}_1 + L & H \\ G & F \end{bmatrix} \begin{bmatrix} x \\ \xi \end{bmatrix} + \begin{bmatrix} \hat{B}_1 \\ 0 \end{bmatrix} u_1 \tag{17a}$$

$$y_1 = \begin{bmatrix} \hat{C}_1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \xi \end{bmatrix} \tag{17b}$$

is a design model for the first control agent. At this stage any uncertainties in the first subsystem dynamics can also be incorporated in this model and a robust controller for the first subsystem can be designed. A similar approach can be undertaken at the second station as well. By such an approach stability and good performance of the resulting closed-loop system can be guaranteed.

VI. CONCLUSION

Representation of uncertain actuator dynamics has been discussed within the framework of [20]. It has been shown that, under mild conditions on uncertain actuator dynamics, it is possible to obtain a rational TFM, possibly dependent on a parameter vector which varies over a subset of a finite dimensional real vector space, to represent such dynamics. Once such a TFM is obtained, a state space realization of it can be combined with the system model and a controller can be designed to guarantee stability and good performance. Note that, the system model itself may also be subject to parameter variations. It has also been shown that the proposed approach has a natural extension to decentralized control.

Decentralized robust controller design has also been discussed within the same framework. It has been demonstrated that local controllers which guarantee overall stability and desired performance can be designed, if certain limited information about rest of the closed-loop system is available to each local agent. Although, for notational simplicity, only interconnected systems with two subsystems has been considered, it is possible to extend the approach to systems with more subsystems by modifying (16) to include the remaining subsystems and corresponding local controllers.

REFERENCES

[1] J. C. Doyle and G. Stein, "Multivariable feedback design: Concepts for a classical/modern synthesis," *IEEE Transactions on Automatic Control*, vol. AC-26, pp. 4-16, 1981.

[2] M. G. Safonov, A. J. Laub, and G. L. Hartmann, "Feedback properties of multivariable systems: the role and use of return difference matrix," *IEEE Transactions on Automatic Control*, vol. AC-26, pp. 47-65, 1981.

[3] J. B. Cruz, J. S. Freudenberg, and D. P. Looze, "A relationship between sensitivity and stability of multivariable feddback systems," *IEEE Transactions on Automatic Control*, vol. AC-26, pp. 66-74, 1981.

[4] I. M. Horowitz, "Quantitative feedback theory," *Proceedings of the IEE, Part-D*, vol. 129, pp. 215-226, 1982.

[5] A. Sideris and M. G. Safonov, "A design algorithm for the robust synthesis of SISO feedback control systems using conformal maps and H_∞-theory," in *Proceedings of the American Control Conference*, pp. 1710-1715, Seattle, WA, June 1986.

[6] G. Stein and M. Athans, "The LQG/LTR procedure for multivariable feedback control design," *IEEE Transactions on Automatic Control*, vol. AC-32, pp. 105-114, 1987.

[7] A. İftar and Ü. Özgüner, "Local LQG/LTR controller design for decentralized systems," *IEEE Transactions on Automatic Control*, vol. AC-32, pp. 926-930, 1987.

[8] R. V. Patel, M. Toda, and B. Sridhar, "Robustness of linear quadratic state feedback designs in the presence of system uncertainty," *IEEE Transactions on Automatic Control*, vol. AC-22, pp. 945-949, 1977.

[9] R. K. Yedavalli, "Perturbation bounds for robust stability in linear state space models," *International Journal of Control*, vol. 42, pp. 1507-1517, 1985.

[10] K. Zhou and P. P. Khargonekar, "Stability robustness bounds for linear state-space models with structured uncertainty," *IEEE Transactions on Automatic Control*, vol. AC-32, pp. 621-623, 1987.

[11] L. H. Keel, S. P. Bhattacharyya, and J. W. Howze, "Robust control with structured perturbations," *IEEE Transactions on Automatic Control*, vol. AC–33, pp. 68–78, 1988.

[12] L. Qiu and E. J. Davison, "New perturbation bounds for the robust stability of linear state space models," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 751–755, Athens, Greece, December 1986.

[13] D. D. Šiljak, "Absolute stability and parameter sensitivity," *International Journal of Control*, vol. 8, pp. 279–283, 1968.

[14] B. R. Barmish, "Invariance of strict Hurwitz property for polynomials with perturbed coefficients," *IEEE Transactions on Automatic Control*, vol. AC–29, pp. 935–936, 1984.

[15] V. L. Kharitonov, "Asyptotic stability of an equilibrium position of a family of systems of linear differential equations," *Differentsialjnia Uravnenia*, vol. 14, pp. 2086–2088, 1978.

[16] J. S. Karmarkar and D. D. Šiljak, "A computer aided design of robust regulators," in *Proceedings of the IFAC Workshop*, pp. 49–58, Denver, CO, June 1979.

[17] J. Ackermann, "Parameter space design of robust control systems," *IEEE Transactions on Automatic Control*, vol. AC–25, pp. 1058–1072, 1980.

[18] R. K. Yedavalli, "Dynamic compensator design for robust stability of linear uncertain systems," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 34–36, Athens, Greece, December 1986.

[19] K. H. Wei and R. K. Yedavalli, "Robust stabilizability for systems with both parameter variation and unstructured uncertainty," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 2082–2087, Los Angeles, CA, December 1987.

[20] A. İftar and Ü. Özgüner, "Structured modeling of unstructured uncertainties and robust controller design in state space," in *Proceedings of the Winter Annual Meeting of ASME*, Chicago, IL, November 1988.

[21] K. D. Young, "A study in actuator/sensor modeling for integrated structural control," Technical Report, Lawrence Livermore National Laboratory, 1987.

[22] A. İftar and Ü. Özgüner, "Decentralized LQG/LTR controller design for interconnected systems," in *Proceedings of the American Control Conference*, pp. 1682–1687, Minneapolis, MN, June 1987.

TIME DOMAIN CONTROL DESIGN FOR ROBUST STABILITY OF DISCRETE-TIME LINEAR REGULATORS

SRI R. KOLLA and JAMES B. FARISON
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606-3390

ABSTRACT

Time domain stability robustness analysis and design for linear multivariable discrete-time systems with bounded uncertainties are the themes of this paper. A state feedback control design algorithm is presented for Linear Quadratic (LQ) regulators, using a quantitative measure called the Stability Robustness Index. These results are based on a recently developed bound on elemental (structured) time-varying perturbations of an asymptotically stable linear time-invariant discrete-time system. The extension of the algorithm to stochastic systems with the Kalman filter as the state estimator (LQG regulators) is described. Simple examples illustrate these new results.

I. INTRODUCTION

The analysis and design of controllers for robust stability of linear systems have been active areas of research for some time. There is considerable literature on these topics for continuous-time systems [1]. Recently, robust stability methods for discrete-time systems have also received interest. In particular, robustness aspects of linear regulators have received considerable attention.

For discrete-time LQ regulators, using the frequency domain approach, Sofonov [2] and Shaked [3] investigated the stability margins. For LQG regulators, the aspect of recovering the LQ robustness properties is addressed by Maciejowski [4] and Ishihara and Takeda [5]. In the time domain, Lee and Lee [6] established the bounds for allowable nonlinear perturbations of discrete-time LQ regulators such that the resulting closed-loop system remains stable, following the method for the continuous-time LQ regulator counterpart [7]. Other important robust control design methods for discrete-time systems are the parametric space design of Ackermann [8] and ultimate boundedness control of Corless and Manela [9].

In this present paper stability robustness is viewed in terms of bounds on the perturbations in the system state-variable model matrices to maintain stability [10]. A recently developed structured perturbation bound for discrete-time systems [11] is used to design linear regulators for robust stability. The notation is defined in Section II. A brief review of the stability bound is presented in Section III. Section IV is devoted to the design of full-state feedback LQ regulators for robust stability. In Section V the design algorithm is extended to stochastic systems with state estimate feedback. Section VI offers some concluding remarks.

II. NOTATION

The singular values of any matrix L are denoted $\sigma(L)$, and defined $\sigma(L) = [\lambda(LL^T)]^{1/2}$, with $\sigma_{max}(L)$ the largest and $\sigma_{min}(L)$ the smallest singular values of L. The notation $|L|$ represents the matrix whose elements are the magnitudes of the elements l_{ij} of L. The symmetric matrix L_s is the symmetric part of a square matrix L: $L_s = (L + L^T)/2$. $L > 0$ denotes positive definite. The notation $\sqrt{L}$ means $\sqrt{L}\sqrt{L}^T = L$.

III. REVIEW OF STABILITY ROBUSTNESS BOUND FOR DISCRETE-TIME SYSTEMS

In this section we briefly review a perturbation bound for robust stability for structured perturbations [11].

Consider the linear discrete-time system described by the difference equation

$$x(k+1) = [A+E(k)]x(k) \tag{1}$$

where x is the n-dimensional state vector, A is an nxn time-invariant, asymptotically stable matrix, and E(k) is an nxn time-varying error (perturbation) matrix. Define constants ϵ_{ij} and ϵ such that the elements $e_{ij}(k)$ of E(k) satisfy

$$e_{ij}(k) \leqslant |e_{ij}(k)|_{max} = \epsilon_{ij} \quad \text{and} \quad \epsilon = \max \epsilon_{ij}. \tag{2}$$

Let $U = [u_{ij}]$, $u_{ij} = \epsilon_{ij}/\epsilon$. Note that $0 \leqslant u_{ij} \leqslant 1$, with $u_{ij} = 0$ if the perturbation $e_{ij}(k)$ of a_{ij} is known to be zero. The following gives a sufficient condition for stability [11].

Theorem 1: The discrete-time system (1) is stable if

$$\epsilon < -\frac{\sigma_{max}(U^T|PA|)_s}{\sigma_{max}(U^T|P|U)} + \sqrt{\left[\frac{\sigma_{max}(U^T|PA|)_s}{\sigma_{max}(U^T|P|U)}\right]^2 + \frac{\sigma_{min}(Q)}{\sigma_{max}(U^T|P|U)}} \tag{3}$$

where P satisfies the discrete-time Lyapunov equation

$$A^TPA - P + Q = 0. \tag{4}$$

If the perturbation e_{ij} of some a_{ij} is not explicitly known, one can take u_{ij} as some positive number, for instance 1, to get an idea of the stable perturbation range by computing the RHS of (3).

IV. CONTROL DESIGN FOR ROBUST STABILITY IN DISCRETE-TIME LQ REGULATORS

Consider the linear, time-invariant discrete-time system described by

$$x(k+1) = Ax(k) + Bu(k), \qquad x(0) = x_0 \tag{5}$$

where x is the n-dimensional state vector and u is the m-dimensional control vector. A and B are constant matrices of appropriate dimensions. Let the performance index to be minimized for the system (5) be

$$J = \sum_{k=0}^{\infty} [x^T(k)\tilde{Q}x(k) + \rho u^T(k)Ru(k)] \tag{6}$$

where $\tilde{Q} > 0$ and $R > 0$ are of appropriate dimensions and ρ is a scalar variable used for design. The matrix triple $(A, B, \sqrt{\tilde{Q}})$ is assumed to be completely controllable and observable.

The optimal control law for this LQ regulator problem [12] is given by

$$u(k) = Gx(k) \tag{7a}$$

where G is the constant gain matrix given by

$$G = -(\rho R + B^TKB)^{-1}B^TKA. \tag{7b}$$

The symmetric matrix K is the positive definite solution of the algebraic discrete-time Riccati equation

$$K = \tilde{Q} + A^TKA - A^TKB(\rho R+B^TKB)^{-1}B^TKA. \tag{8}$$

It is well known that the closed-loop system matrix $A_c = A+BG$ is asymptotically stable [12].

Let ΔA, ΔB and ΔG be the perturbation matrices of matrices A, B and G, respectively. Then the perturbed closed-loop system matrix is

$$\begin{aligned} A_{cp} &= (A+\Delta A) + (B+\Delta B)(G+\Delta G) \\ &= (A+BG) + [\Delta A + (\Delta B)G + (B+\Delta B)\Delta G] \\ &= A_c + E(k) \end{aligned} \tag{9a}$$

where

$$E = \Delta A + (\Delta B)G + B(\Delta G) + (\Delta B)(\Delta G). \tag{9b}$$

Let $\overline{\Delta A} = |\Delta A(k)|_{max}$, etc., and define

$$\tilde{E} = \overline{\Delta A} + (\overline{\Delta B})|G| + |B|(\overline{\Delta G}) + (\overline{\Delta B})(\overline{\Delta G}). \tag{10}$$

With $\tilde{\epsilon} = \max \tilde{e}_{ij}$, $\epsilon \leqslant \tilde{\epsilon}$.

Design Observation 1: The perturbed LQ regulator system is stable for all perturbations in A, B and G, in the sense of

(3), if [10]

$$\tilde{\epsilon} < \mu \tag{11a}$$

where μ is given by

$$\mu = -\frac{\sigma_{\max}(U^T|PA_c|)_s}{\sigma_{\max}(U^T|P|U)} + \sqrt{\left[\frac{\sigma_{\max}(U^T|PA_c|)_s}{\sigma_{\max}(U^T|P|U)}\right]^2 + \frac{\sigma_{\min}(Q)}{\sigma_{\max}(U^T|P|U)}} \tag{11b}$$

and P satisfies the Lyapunov equation

$$A_c^T P A_c - P + Q = 0. \tag{11c}$$

This result is less conservative if one can use ϵ in (11a) instead of $\tilde{\epsilon}$. However, it is not always possible to calculate ϵ.

It is clear from (10) and (11) that both $\tilde{\epsilon}$ and μ are functions of the control gain G. It is, therefore, appropriate to consider the relative difference of these two parameters in comparing and designing different controllers (6)-(8) from the stability robustness point of view. To this end, as a measure of stability robustness, we define the Stability Robustness Index

$$\beta = \mu - \tilde{\epsilon}. \tag{12}$$

Thus, $\beta > 0$ corresponds to the stability robustness region (11a). In practice the following situations may arise.

Case 1: The maximum-modulus perturbation matrices $\overline{\Delta A}$, $\overline{\Delta B}$ and $\overline{\Delta G}$, and therefore $\tilde{E}$ and $\tilde{e}_{ij}$ are known. For this case the U matrix in (11b) is formed by $u_{ij} = \tilde{e}_{ij}/\tilde{\epsilon}$, and G assures stability if $\beta = \mu - \tilde{\epsilon} > 0$.

Case 2: Information on the perturbation matrices is not completely known. For this case one can take $u_{ij} = 1$ for the positions with unknown perturbation. Now $\tilde{\epsilon}$ (and, therefore, β) is not known. One might choose G to maximize μ (although the unknown $\tilde{\epsilon}$ also depends on G).

Example 1: Consider the linear uncertain discrete-time system

$$x(k+1) = \begin{pmatrix} 0 & 1 \\ a_1 & a_2 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k)$$

where

$$-1.5 \leqslant a_1(k) \leqslant -0.5$$

$$-2.4 \leqslant a_2(k) \leqslant -1.6.$$

This system can be represented as

$$x(k+1) = (A+\Delta A)x(k) + (B+\Delta B)u(k)$$

where $A = \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\overline{\Delta A} = \begin{pmatrix} 0.0 & 0.0 \\ 0.5 & 0.4 \end{pmatrix}$ and $\overline{\Delta B} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

The nominal A and some of the perturbation range are unstable. Assuming $\Delta G = 0$,

$$\widetilde{E} = \begin{pmatrix} 0.0 & 0.0 \\ 0.5 & 0.4 \end{pmatrix}.$$

As the $\widetilde{E}$ matrix is completely known, this system can be treated as Case 1. With $\widetilde{\epsilon} = 0.5$,

$$U = \begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 0.8 \end{pmatrix}.$$

Taking $\widetilde{Q} = I$ and $R = I$ in the LQ regulator design and varying ρ, different stabilizing control gains can be obtained for the nominal system. For $\rho = 0.01$, (7) gives

$$G = [0.9951 \quad 1.9807].$$

With this gain matrix, and $Q = I$, (11) gives

$$\mu = 0.5383$$

and the Stability Robustness Index (12) is

$$\beta = 0.0383 > 0.$$

The designed gain matrix, therefore, stabilizes the uncertain system for all given perturbations.

Example 2: Consider the linear discrete-time system

$$x(k+1) = \begin{pmatrix} 0 & 1 \\ a_1 & a_2 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k).$$

Let a_1 and a_2 be varying with nominal values $a_1 = -0.5$ and $a_2 = -1.0$. Considering this problem as Case 2,

$$U = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

Again taking $\widetilde{Q} = I$ and $R = I$ in the LQ regulator design, and $Q = I$ in (11), the variation of μ for different values of ρ is shown in Fig. 1. By (11a), the system remains stable for variations satisfying $\widetilde{\epsilon} < \mu$.

From the asymptotic properties of the general LQ regulator [12], both closed-loop poles reach the origin as $\rho \rightarrow 0$ and are the same as the open-loop poles as $\rho \rightarrow \infty$. From Fig. 1 it is clear that as $\rho \rightarrow 0$, μ for the closed-loop LQ regulator system approaches 0.5 and, as $\rho \rightarrow \infty$, μ for the closed-loop system approaches the open-loop system value (= 0.0830).

Remark 1: Recently, Katayama and Sasaki [13] showed that "matched" continuous-time LQ regulator systems have several important properties. In particular, from their analysis it can be said that, for the continuous-time LQ regulator design for the matched continuous-time system similar to our Example 2, elements a_1 and a_2 can tolerate arbitrarily large perturbations. Alternatively, it can be shown that the continuous-time system perturbation bound [10] for that LQ regulator problem approaches ∞ as $\rho \rightarrow 0$. In other words, that "matched" continuous-time LQ regulator system can tolerate arbitrarily large perturbations in a_1 and a_2. However, from Example 2, it can be shown that "matched" discrete-time LQ regulator systems can not always tolerate arbitrarily large perturbations ($\epsilon > 0.5$ causes instability with $\rho \rightarrow 0$).

V. EXTENSION TO LQG REGULATORS

In this section, we extend the design method presented in the last section to stochastic systems with the Kalman filter as the state estimator in the linear regulator (LQG regulator).

Consider the linear time-invariant discrete-time stochastic system

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{13a}$$

$$z(k) = Mx(k) + v(k) \tag{13b}$$

where x is the n-dimensional state vector, u is the m-dimensional control vector, z is the l-dimensional measurement vector, and w and v are the system and measurement noise vectors, respectively. Constant matrices A, B and M are of appropriate dimensions. The initial condition x(0) is assumed to be a zero-mean Gaussian random vector with variance X_0,

$$E[x(0)] = 0, \quad E[x(0)x^T(0)] = X_0. \tag{14}$$

The system and measurement noise vectors are assumed to be zero-mean Gaussian white-noise processes with

$$E[w(k)] = 0, \quad E[v(k)] = 0 \tag{15a}$$

and

$$E\left[\begin{bmatrix} w(j) \\ v(j) \end{bmatrix} \begin{bmatrix} w^T(k) & v^T(k) \end{bmatrix}\right] = \begin{bmatrix} W & 0 \\ 0 & V \end{bmatrix} \delta_{jk} \tag{15b}$$

with $W > 0$ and $V > 0$. We wish to find the control $u(k)$ that minimizes the performance index

$$J = \lim_{N\to\infty} \frac{1}{N} E\left[\sum_{k=0}^{N-1} [x^T(k)\widetilde{Q}x(k) + u^T(k)Ru(k)]\right] \tag{16}$$

where $\widetilde{Q} > 0$ and $R > 0$ are of appropriate dimensions. We assume that the matrix pairs (A, B) and $(A, \sqrt{W})$ are completely controllable and the pairs $(A, \sqrt{\widetilde{Q}})$ and (A, M) are completely observable.

The optimal control law for this LQG regulator [12] is given by

$$u(k) = G\hat{x}(k) \tag{17a}$$

where

$$G = -(R+B^TKB)^{-1}B^TKA, \tag{17b}$$

and $\hat{x}(k)$ is obtained from the Kalman filter

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + \hat{G}[z(k)-M\hat{x}(k)] \tag{17c}$$

and

$$\hat{G} = A\widetilde{P}M^T(V+M\widetilde{P}M^T)^{-1}. \tag{17d}$$

K and $\widetilde{P}$ satisfy the algebraic discrete Riccati matrix equations

$$K = \widetilde{Q} + A^TKA - A^TKB(R+B^TKB)^{-1}B^TKA \tag{18a}$$

$$\widetilde{P} = W + A\widetilde{P}A^T - A\widetilde{P}M^T(V+M\widetilde{P}M^T)^{-1}M\widetilde{P}A^T. \tag{18b}$$

The closed-loop LQG regulator system can be written as

$$\begin{vmatrix} x(k+1) \\ \hat{x}(k+1) \end{vmatrix} = \begin{vmatrix} A & BG \\ \hat{G}M & \hat{A}_c \end{vmatrix} \begin{vmatrix} x(k) \\ \hat{x}(k) \end{vmatrix} + \begin{vmatrix} I & 0 \\ 0 & \hat{G} \end{vmatrix} \begin{vmatrix} w(k) \\ v(k) \end{vmatrix} \tag{19}$$

where $\hat{A}_c = A+BG-\hat{G}M$. It is well known that, if the system satisfies the controllability and observability conditions described before, the closed-loop LQG regulator system is asymptotically stable [12].

Now, let ΔA, ΔB, ΔM, ΔG and $\Delta\hat{G}$ be the perturbation matrices of matrices A, B, M, G and $\hat{G}$, respectively. Then the perturbed closed-loop system matrix is

$$\begin{aligned} A_{cp} &= \begin{vmatrix} (A+\Delta A) & (B+\Delta B)(G+\Delta G) \\ (\hat{G}+\Delta\hat{G})(M+\Delta M) & (\hat{A}_c+\Delta\hat{A}_c) \end{vmatrix} \\ &= \begin{vmatrix} A & BG \\ \hat{G}M & \hat{A}_c \end{vmatrix} + \begin{vmatrix} \Delta A & \Delta BG+(B+\Delta B)\Delta G \\ \hat{G}\Delta M+\Delta\hat{G}(M+\Delta M) & \Delta\hat{A}_c \end{vmatrix} \\ &= A_c + E(k) \end{aligned} \tag{20}$$

where $\Delta\hat{A}_c = [\Delta A + (B+\Delta B)\Delta G + (\Delta B)G - \Delta\hat{G}(M+\Delta M) - \hat{G}(\Delta M)]$. Define

$$\widetilde{E} = \begin{bmatrix} \overline{\Delta A} & (\overline{\Delta B})|G|+|B|(\overline{\Delta G})+(\overline{\Delta B})(\overline{\Delta G}) \\ |\hat{G}|(\overline{\Delta M})+(\overline{\Delta\hat{G}})|M|+(\overline{\Delta\hat{G}})(\overline{\Delta M}) & \overline{\Delta\hat{A}_c} \end{bmatrix} \qquad (21)$$

where $\overline{\Delta\hat{A}_c} = \overline{\Delta A} + |B|(\overline{\Delta G}) + (\overline{\Delta B})(\overline{\Delta G}) + (\overline{\Delta B})|G| + (\overline{\Delta\hat{G}})|M| + (\overline{\Delta\hat{G}})(\overline{\Delta M}) + |\hat{G}|(\overline{\Delta M})$. We can now make a design observation for robust stability of LQG regulator systems similar to that of Observation 1 for LQ regulator systems. The robust control design procedure described for LQ regulators can be extended for LQG regulators.

Design Observation 2: The perturbed LQG regulator system is stable for all perturbations in A, B, M, G, and $\hat{G}$, in the sense of (3), if [10]

$$\widetilde{\epsilon} < \mu \qquad (22)$$

where μ is given by the form (11b) with A_c in (20).

Example 3: Consider the system

$$x(k+1) = \begin{pmatrix} 0 & 1 \\ -0.5 & -1.0 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k) + w(k)$$

$$z(k) = [1 \quad 0]x(k) + v(k)$$

with W = I and V = 1.0. Taking $\widetilde{Q} = I$ and R = I in the performance index (16),

$$A_c = \begin{bmatrix} 0.0 & 1.0 & 0.0 & 0.0 \\ -0.5 & -1.0 & 0.3653 & 0.6439 \\ -0.4365 & 0.0 & 0.4365 & 1.0 \\ 0.0635 & 0.0 & -0.1982 & -0.3561 \end{bmatrix}.$$

Assuming that the a_{21} and a_{22} elements of the A matrices in both the system and the estimator are subjected to independent and unknown variations (Case 2),

$$U = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

For this U, and Q = I in (11), the stability robustness bound μ for the above closed-loop system matrix is

$$\mu = 0.0688.$$

This closed-loop LQG regulator system remains stable for all perturbations that satisfy $\tilde{\epsilon} < \mu$.

VI. CONCLUSIONS

In this paper, an algorithm for designing controllers for stability robustness of discrete-time linear regulators in the time-domain is presented. To assure stability under parameter perturbations, the control gain matrix G can be varied (through $\tilde{Q}$ and R) to obtain a result in the stability robustness region. The design algorithm uses a Stability Robustness Index based on a recently developed time-varying perturbation bound. Both deterministic LQ regulators and stochastic LQG regulators are considered. In the case of matched LQ regulator systems, important differences between continuous-time and discrete-time are noted. The method presented in this paper can be extended to design reduced-order dynamic compensators for robust stability.

REFERENCES

1. P. Dorato, "A historical review of robust control," IEEE Control System Magazine, vol. 7, pp. 44-47, April 1987.

2. M. G. Sofonov, Stability and Robustness of Multivariable Feedback Systems. Cambridge, MA: M.I.T. Press, 1980.

3. U. Shaked, "Guaranteed stability margins for the discrete-time linear quadratic regulators," IEEE Trans. Automat. Contr., vol. AC-31, pp. 162-165, February 1986.

4. J. M. Maciejowski, "Asymptotic recovery for discrete-time systems," IEEE Trans. Automat. Contr., vol. AC-30, pp. 602-605, June 1985.

5. T. Ishihara and H. Takeda, "Loop transfer recovery technique for discrete-time optimal regulators using prediction estimators," IEEE Trans. Automat. Contr., vol. AC-31, pp. 1149-1151, December 1986.

6. T.-T Lee and S.-H Lee, "Discrete optimal control with eigenvalues assigned inside a circular region," IEEE Trans. Automat. Contr., vol. AC-31, pp. 958-962, October 1986.

7. R. V. Patel, M. Toda and B. Sridhar, "Robustness of linear quadratic state feedback designs in the presence of system uncertainty," IEEE Trans. Automat. Contr., vol. AC-22, pp. 945-949, December 1977.

8. J. Ackermann, "Parameter space design of robust control systems," IEEE Trans. Automat. Contr., vol. AC-25, pp. 1058-1072, December 1980.

9. M. Corless and J. Manela, "Control of uncertain discrete-time systems," Proc. Amer. Contr. Conf., Seattle, WA, June 18-20, 1986.

10. R. K. Yedavalli, S. S. Banda and D. B. Ridgely, "Time-domain stability robustness measures for linear regulators," *J. Guidance Contr. and Dynamics*, vol. 8, pp. 520-524, 1985.

11. S. R. Kolla, J. B. Farison and R. K. Yedavalli "State-space model perturbation bounds for robust stability of linear discrete-time systems," *31 st Midwest Symposium on Circuits and Systems*, St. Louis, MO, August 10-12, 1988.

12. H. Kwakernaak and R. Sivan, *Linear Optimal Control*. New York: Wiley-Interscience, 1972.

13. T. Katayama and S. Sasaki, "Robust stability of linear quadratic state feedback regulator under system uncertainty," *Int. J. Contr.*, vol. 45, pp. 391-405, 1987.

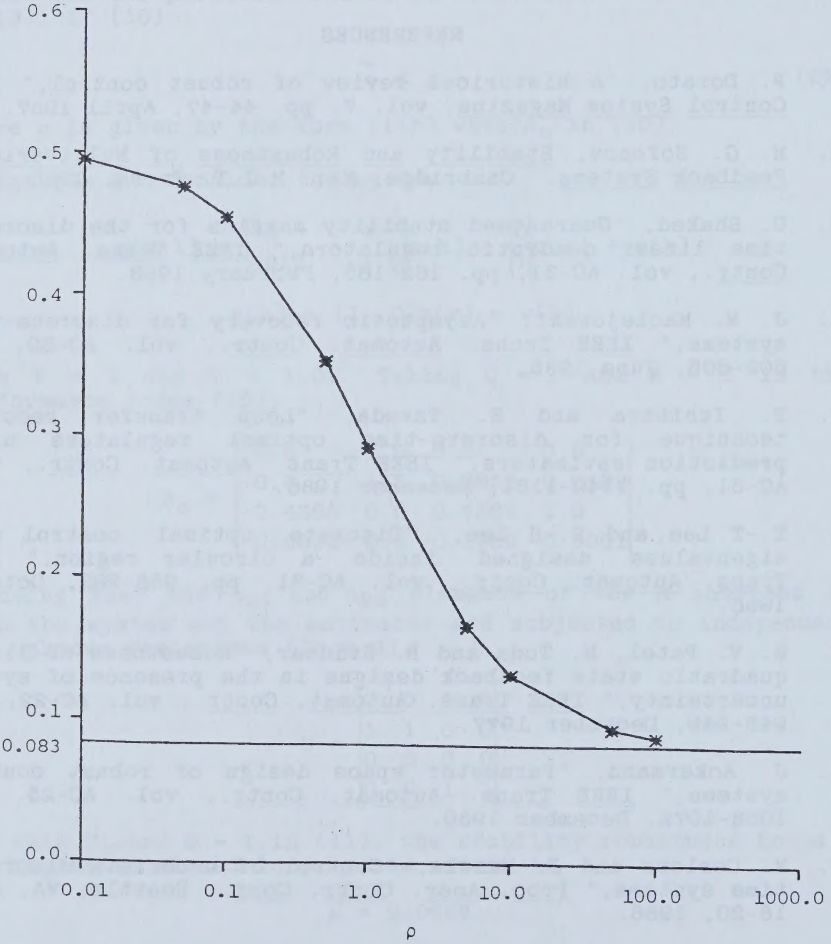


Fig. 1. Variation of μ for different values of ρ.

SPECTRAL APPROXIMATION OF ALMOST BLOCK-TRIANGULAR OPERATORS AND HANKEL-NORM APPROXIMATION OF TWO FREQUENCY SCALE TRANSFER MATRICES

HOSSEIN OLOOMI AND MAHMOUD E. SAWAN
EE Dept., The Wichita State University, Wichita, KS 67208

ABSTRACT

Analytic perturbation theory is used to solve the eigenvalue problem for almost block-triangular operators on C^n. Results are shown to be useful in approximating the Hankel-norm of two frequency scale transfer matrices (TFSTMs).

PROBLEM FORMULATION

Using the notation used in [2], we address the solution of the following eigenvalue problem.

(A) Find $\lambda(\varepsilon)\varepsilon C$, $0\neq\phi(\varepsilon)\varepsilon C^n$ such that $W(\varepsilon)\phi(\varepsilon) = \lambda(\varepsilon)\phi(\varepsilon)$ where

$$W(\varepsilon) = \begin{bmatrix} W_{aa}(\varepsilon) & \varepsilon\tilde{W}_{ab}(\varepsilon) \\ W_{ba}(\varepsilon) & W_{bb}(\varepsilon) \end{bmatrix} \varepsilon C^{n\times n}$$

is an almost block-triangular operator analytic in the small parameter $\varepsilon\,\varepsilon[0,\varepsilon_0]$.

Setting $W_{aa}(\varepsilon) := W_{aa}(0) + \varepsilon\tilde{W}_{aa}(\varepsilon)$, $W_{bb}(\varepsilon) := W_{bb}(0) + \varepsilon\tilde{W}_{bb}(\varepsilon)$, and $W(\varepsilon) := W_L + \varepsilon\tilde{W}(\varepsilon)$, we have the following approximate eigenvalue problem.

(A') Find $\lambda_0\varepsilon C$, $0\neq\phi\varepsilon C^n$ such that $W_L\phi = \lambda_0\phi$ where

$$W_L := \begin{bmatrix} W_{aa}(0) & 0 \\ W_{ab}(0) & W_{bb}(0) \end{bmatrix}$$

is a block-triangular part of $W(\varepsilon)$ and is independent of ε, and $W_{aa}(0)\,\varepsilon\, C^{n_1\times n_1}$, $W_{bb}(0)\,\varepsilon\, C^{n_2\times n_2}$.

Theorem 1 shows that the solution to (A') is a neighboring solution to (A).

THEOREM 1: Let $\lambda_0\varepsilon\sigma(W_L)$ be simple and $\phi_0\varepsilon C^n$ be the corresponding right eigenvector. For sufficiently small ε, there exists a simple eigenvalue $\lambda(\varepsilon)\varepsilon\ \sigma(W(\varepsilon))$ and the corresponding right eigenvector $\phi(\varepsilon)$ such that

$$\lambda(\varepsilon) = \lambda_0 + O(\varepsilon) \quad ; \quad \phi(\varepsilon) = \phi_0 + O(\varepsilon).$$

Furthermore, define

$$\phi_0 := \begin{bmatrix} \phi_{10} \\ \phi_{20} \end{bmatrix}, \quad \phi_{10}\varepsilon C^{n_1}, \quad \phi_{20}\varepsilon C^{n_2}, \quad n_1 + n_2 = n.$$

i) If $\lambda_0\varepsilon\sigma(W_{aa}(0))$, then

$$\phi_0 = \begin{bmatrix} I_{n_1} \\ [\lambda_0 I_{n_2} - W_{bb}(0)]^{-1}W_{ba}(0) \end{bmatrix} \phi_{10}$$

where (λ_0, ϕ_{10}) is the right eigenelement of $W_{aa}(0)$.

ii) If $\lambda_0\varepsilon\sigma(W_{bb}(0))$, then

$$\phi_0 = \begin{bmatrix} 0 \\ I_{n_2} \end{bmatrix} \phi_{20}$$

where (λ_0, ϕ_{20}) is the right eigenelement of $W_{bb}(0)$.

As a consequence of this theorem, the Hankel norm of TFSTMs can be approximated. To this end, let $H(s,\varepsilon)\varepsilon F_{\varepsilon}(s)^{p\times q}$ have the singularly perturbed analytic state space realization [4]

$$H(s,\varepsilon) := \left(\begin{array}{cc|c} A_{11}(\varepsilon) & A_{12}(\varepsilon) & B_1(\varepsilon) \\ A_{21}(\varepsilon)/\varepsilon & A_{22}(\varepsilon)/\varepsilon & B_2(\varepsilon)/\varepsilon \\ \hline C_1(\varepsilon) & C_2(\varepsilon) & D(\varepsilon) \end{array}\right) := \left(\begin{array}{c|c} A(\varepsilon) & B(\varepsilon) \\ \hline C(\varepsilon) & D(\varepsilon) \end{array}\right).$$

If $H(s,\varepsilon)$ is Hurwitz, the controllability and observability gramians defined by

$$W_c(\varepsilon) := \int_0^{\infty} \exp[A(\varepsilon)t]B(\varepsilon)B^T(\varepsilon)\exp[A^T(\varepsilon)t]dt ,$$

$$W_o(\varepsilon) := \int_0^{\infty} \exp[A^T(\varepsilon)t]C^T(\varepsilon)C(\varepsilon)\exp[A(\varepsilon)t]dt$$

satisfy the algebraic Lyapunov equations [3]

$$A(\varepsilon)W_c(\varepsilon) + W_c(\varepsilon)A^T(\varepsilon) + B(\varepsilon)B^T(\varepsilon) = 0 ,$$

$$A^T(\varepsilon)W_o(\varepsilon) + W_o(\varepsilon)A(\varepsilon) + C^T(\varepsilon)C(\varepsilon) = 0.$$

It is well-known that these gramians take the following two forms

$$W_c(\varepsilon)=\begin{bmatrix} W_c^{11}(\varepsilon) & W_c^{12}(\varepsilon) \\ W_c^{12}(\varepsilon)^T & W_c^{22}(\varepsilon)/\varepsilon \end{bmatrix} , \qquad W_o(\varepsilon)=\begin{bmatrix} W_o^{11}(\varepsilon) & \varepsilon W_o^{12}(\varepsilon) \\ \varepsilon W_o^{12}(\varepsilon)^T & \varepsilon W_o^{22}(\varepsilon) \end{bmatrix}.$$

Dropping the arguments for convenience, observe that

$$W_{co}(\varepsilon) := W_c(\varepsilon)W_o(\varepsilon) = \begin{bmatrix} W_c^{11}W_o^{11} + \varepsilon W_c^{12}W_o^{12T} & \varepsilon[\, W_c^{11}W_o^{12} + W_c^{12}W_o^{22} \,] \\ W_c^{12T}W_o^{11} + W_c^{22}W_o^{12T} & W_c^{22}W_o^{22} + \varepsilon W_c^{12T}W_o^{12T} \end{bmatrix}$$

has an almost block-triangular structure.

Since $\det(A_{22}(0)) \neq 0$, standard results can be used to obtain the realizations of the slow and fast subsystems. It can be shown that

$$W_{co}^s = W_c^{11}(0)W_o^{11}(0) \qquad , \qquad W_{co}^f = W_c^{22}(0)W_o^{22}(0)$$

are the products of the gramians of these subsystems, respectively. Thus

THEOREM 2: Let $H(s,\varepsilon)$ have the singularly perturbed analytic state space realization as given. Let

i) $\mathrm{Re}\lambda(A_{11}(0)-A_{12}(0)A_{22}^{-1}(0)A_{21}(0)) < 0$, $\mathrm{Re}\lambda(A_{22}(0)) < 0$, and

ii) $\gamma^2 = \max\{\lambda_{\max}(W_{co}^s),\ \lambda_{\max}(W_{co}^f)\}$ be the unique maximal element of $\sigma(W_{co}^s) \cup \sigma(W_{co}^f)$.

Then the Hankle norm of $H(s,\varepsilon)$ can be approximated by $\|H(s,\varepsilon)\|_H - \gamma = O(\varepsilon)$.

REFERENCES

[1] T. Kato, "Perturbation Theory for Linear Operators," Berlin, Germany: Springer-Verlag, 1966.

[2] F. Chatelin, "Spectral Approximation of Linear Operators," Academic Press, 1983.

[3] K. Glover, "All Optimal Hankel-norm Approximation for Linear Multivariable Systems and Their L^{∞} -error bound," Int. J. Contr., Vol.39, pp. 1115-1193, 1984.

[4] D.W. Luse, "State-Space Realization of Multiple-Frequency-Scale Transfer Matrice," IEEE Trans. Automat. Contr., Vol.33, No.2, pp. 185-187, Feb. 1988.

BLOCK TRIANGULARIZATION OF DISCRETIZED MULTIPARAMETER SINGULARLY PERTURBED SYSTEMS

GEORGE L. SYRMOS AND EYAD H. ABED
Department of Electrical Engineering
and the Systems Research Center
University of Maryland, College Park, MD 20742, USA

Block triangularization of discrete-time singularly perturbed systems involving several independent small parameters is considered. We focus on a model resulting from time discretization of a standard multiparameter singularly perturbed continuous-time model using the backward (or implicit) Euler discretization procedure. The step-size h is taken as a fixed positive quantity, while the singular perturbation parameters ϵ_i are taken to be small and positive, with no constraint placed on their mutual ratios ϵ_i/ϵ_j for all i, j.

Consider the standard multiparameter singularly perturbed continuous-time system which appears as Eq. (1) of [1] (not reproduced here for brevity). There, $x \in R^n$ denotes the "slow" states, $y \in R^m$ the "fast" states, and $\epsilon = (\epsilon_1, ..., \epsilon_M)$ is the vector of singular perturbation parameters, taken small and positive. The matrix $E(\epsilon)$:=block diag$(\epsilon_1 I_{m_1}, ..., \epsilon_m I_{m_M})$ where I_k denotes the $k \times k$ identity matrix. Denote by A_{ij}, $i, j = 1, 2$ the coefficient matrices $A_{11} = A$, $A_{12} = B$, $A_{21} = C$ and $A_{22} = D$ where the notation A, B, C, D is as specified in [1].

The backward Euler discretization procedure applied to system [1, Eq. (1)] results in

$$G(\epsilon, h)\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} I - hA_{11} & -hA_{12} \\ -hE^{-1}(\epsilon)A_{21} & I - hE^{-1}(\epsilon)A_{22} \end{pmatrix}\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix}, \tag{1}$$

where $h > 0$ denotes the step-size, and x_k, y_k denote $x(hk)$, $y(hk)$, respectively, for integers $k \geq 0$ [2].

To block triangularize the matrix $G(\epsilon, h)$, we use a similarity transformation of the form

$$\begin{pmatrix} I - hA_{11} & -hA_{12} \\ -hE^{-1}(\epsilon)A_{21} & I - hE^{-1}(\epsilon)A_{22} \end{pmatrix}\begin{pmatrix} I_n & 0 \\ X & I_m \end{pmatrix}$$

$$= \begin{pmatrix} I_n & 0 \\ X & I_m \end{pmatrix}\begin{pmatrix} Z & -hA_{12} \\ 0 & Y \end{pmatrix}, \tag{2}$$

where X, Y, and Z are matrices of appropriate dimension. These matrices may be determined straightforwardly, under the assumption that A_{22} is nonsingular.

These considerations lead to the following result, given in terms of a matrix $X(\epsilon)$ which is a solution to the associated algebraic matrix Riccati equation $A_{21} + A_{22}X - E(\epsilon)XA_{11} - E(\epsilon)XA_{12}X = 0$.

Theorem 1. Fix $h \neq 0$ and let $||\epsilon||$ be sufficiently small, with $\epsilon_i \neq 0$, $i = 1, ..., M$. Then the spectrum of $G(\epsilon, h)$ is given by

$$\sigma(G(\epsilon,h)) = \sigma(I - hA_{11} - hA_{12}X(\epsilon)) \\ \cup\ \sigma(I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12}). \tag{3}$$

We now make the following assumptions (see [1] for details).

(i) $A_{11} - A_{12}A_{22}^{-1}A_{21}$ is stable in the continuous-time sense (Hurwitz stable), and

(ii) A_{22} is strongly block $\mathcal{D}$-stable.

Under (i) and (ii), it follows that $G(\epsilon,h)$ is invertible for fixed $h > 0$ and all sufficiently small $||\epsilon||$. Therefore, for any $h > 0$, and all sufficiently small $||\epsilon||$, $G^{-1}(\epsilon,h)$ exists. This justifies rewriting system (1) in *forward* time as follows:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = G^{-1}(\epsilon,h)\begin{pmatrix} x_k \\ y_k \end{pmatrix}. \tag{4}$$

In fact, $G^{-1}(\epsilon,h)$ of Eq. (4) is given *explicitly* by the formula

$$G^{-1}(\epsilon,h) =: K(\epsilon,h) = \begin{pmatrix} K_{11}(\epsilon,h) & K_{12}(\epsilon,h) \\ K_{21}(\epsilon,h) & K_{22}(\epsilon,h) \end{pmatrix}, \tag{5}$$

where the K_{ij}, $i,j = 1,2$, may be computed as:

$$K_{11}(\epsilon,h) = (I - hA_{11} - hA_{12}X(\epsilon))^{-1} \\ - h(I - hA_{11} - hA_{12}X(\epsilon))^{-1}A_{12}(I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12})^{-1}X(\epsilon), \tag{6a}$$

$$K_{12}(\epsilon,h) = h(I - hA_{11} - hA_{12}X(\epsilon))^{-1}A_{12}(I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12})^{-1}, \tag{6b}$$

$$K_{21}(\epsilon,h) = X(\epsilon)(I - hA_{11} - hA_{12}X(\epsilon))^{-1} \\ - hX(\epsilon)(I - hA_{11} - hA_{12}X(\epsilon))^{-1}A_{12}(I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12})^{-1}X(\epsilon) \\ + (I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12})^{-1}X(\epsilon), \tag{6c}$$

$$K_{22}(\epsilon,h) = hX(\epsilon)(I - hA_{11} - hA_{12}X(\epsilon))^{-1}A_{12}(I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12})^{-1} \\ + (I - hE^{-1}(\epsilon)A_{22} + hX(\epsilon)A_{12})^{-1}. \tag{6d}$$

Acknowledgement

The authors are grateful to Professor H.K. Khalil for his comments on the earlier paper [2]. The support of the NSF under Grants ECS-86-57561 and CDR-85-00108 is gratefully acknowledged.

References

[1] E.H. Abed, "Strong $\mathcal{D}$-stability ," *Systems and Control Letters*, Vol. 7, pp. 207-212, June 1986.

[2] E.H. Abed, "On multiparameter singularly perturbed discrete-time systems," *Proc. of the 27th IEEE Conference on Decision and Control*, Los Angeles, pp. 2104-2105, Dec. 1987.

AN ADAPTIVE CONTROLLER TO ACHIEVE AN ARBITRARILY GOOD TRANSIENT AND STEADY-STATE RESPONSE

D.E. Miller and E.J. Davison*
Department of Electrical Engineering
University of Toronto
Toronto, Ontario
Canada M5S 1A4

ABSTRACT

In this paper we pose a new model reference adaptive control problem in which the objective is not to force the error between the plant output and the reference model output asymptotically to zero, but instead it is to force the error to be less than an (arbitrarily small) prespecified constant after an (arbitrarily short) prespecified period of time, with an (arbitrarily small) prespecified upper bound on the amount of overshoot. It is shown that to achieve this goal it is sufficient that the plant be minimum phase and that an upper bound on the relative degree be known; knowledge of an upper bound on the plant order and the sign of the high frequency gain are not required. Furthermore, the controller proposed has dynamics of order equal to the upper bound on the relative degree less one.

1. INTRODUCTION

In model reference adaptive control (MRAC), the usual objective is to force the output of a partially known plant to asymptotically track the output of a prespecified reference model for all reference model inputs in a prespecified class, typically the set of all piecewise continuous and bounded functions; the reference model is designed in such a way as to embody the desired behaviour of the controlled plant. In order to achieve this, the four typical assumptions made about a single-input single-output (SISO) plant are that (i) the plant is minimum phase, (ii) an upper bound on the plant order is known, (iii) the relative degree is known, and (iv) the sign of the high-frequency gain is known, e.g. [1], [2]. In [3] it is shown that assumption (iv) is unnecessary, and in [4] it is shown that (iii) can be replaced with the assumption that it is known that the relative degree is either one or two; thus the sign of the high-frequency gain need not be known, and in some circumstances the relative degree need not be known exactly. For a recent monograph on the area of MRAC, see [5].

In order to provide asymptotic error regulation for a rich class of reference model input signals, it is typically necessary to have an adaptive controller of order roughly proportional to the order of the plant, e.g. [1], [2]; this, of course, necessitates assumption (ii) above. However, in practise it is often satisfactory if there is a "small", but not necessarily zero, steady-state error, which raises the following question: can we get away with weaker plant assumptions, and possibly with a less complex controller, if we weaken the asymptotic regulation requirement to an "approximate regulation" condition? Here we consider the case when the class of reference model input signals are the piecewise continuous and bounded functions, and we say that "approximate regulation" occurs if the error is less than an (arbitrary) prespecified constant as $t \rightarrow \infty$. It turns out that it is sufficient to assume that the plant is minimum phase and that an upper bound on the relative degree is known, although no knowledge of the relative degree is required if an unlimited number of dynamic elements are at our disposal; hence, we dispense with assumptions (ii) and (iv), and replace (iii) with a weaker version. Furthermore, the order of the dynamics of the proposed adaptive controller is equal to the upper bound on the relative degree less one, which may be considerably smaller than the plant

---

* This work has been supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. A4396.

order.

In most results on MRAC, e.g. [1], [2], little can be said about the system's closed loop transient response, even though it is an important consideration. This raises the following question: what type of assumptions are required in order to achieve "approximate regulation" *and* a "good transient response"? If we define "approximate regulation" together with a "good transient response" to mean that the error is less than an (arbitrarily small) prespecified constant after an (arbitrarily short) prespecified period of time and that the "overshoot" of the error is less than an (arbitrarily small) prespecified constant, then it turns out that the assumptions which must be made about the plant are exactly those listed above, and that the order of the controller can be made equal to the upper bound on the relative degree less one.

In Section 2 we shall give a formal problem statement, and in Section 3 we shall present adaptive controllers which will solve the problem if the plant is minimum phase and the relative degree is known; it will then be shown that knowledge of the exact relative degree can be replaced by knowledge of an upper bound on the relative degree of the plant. In Section 4 we give concluding remarks.

The following notation is required. Let **N** denote the integers, **R** denote the real numbers, **C** denote the complex numbers, $\mathbf{C}^-$ denote the complex numbers with real parts which are strictly less than zero, and $\mathbf{C}^+$ denote the complex numbers with real parts greater than or equal to zero. With $p \in \mathbf{N}$, let $\mathbf{p}$ denote the set of integers $\{1, 2, \ldots, p\}$. With $x \in \mathbf{R}$, define

$$sgn\, x = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

For $A \in \mathbf{R}^{n \times m}$, we define $\|A\|$ to be the ∞-norm of A, i.e. if we write $A = \{a_{ij}\}$, then $\|A\| = \max_{i \in \mathbf{n}} \sum_{j=1}^{m} |a_{ij}|$. With $A \in \mathbf{R}^{n \times n}$, we say that A is *stable* if the real part of every eigenvalue of A is negative.

Define PC to be the set of $\mathbf{R}^n$-valued piecewise continuous functions defined on $[0, \infty)$. Define PC_∞ to be the set of those elements of PC which are essentially bounded on $[0, \infty)$. Define PC_∞^1 to be the subset of PC_∞ whose elements are continuous and have derivatives in PC_∞; more precisely, $f \in PC_\infty$ is an element of PC_∞^1 if

(i) f is continuous,

(ii) f is differentiable at all but a finite number of points in any finite sub-interval of $[0, \infty)$, and

(iii) the function g defined by

$$g(x) = \begin{cases} \dot{f}(x) & \text{if } f \text{ is differentiable at } x \\ 0 & \text{otherwise} \end{cases}$$

is an element of PC_∞.

In the following, basically the only situation in which $\dot{f}$ (with $f \in PC_\infty^1$) arises is inside an integral; since the value of the integrand on a set of measure zero has no effect on the integral, we can extend the domain of definition of $\dot{f}$ to $[0, \infty)$ by defining it arbitrarily at the points where the derivative does not exist. Finally, if $f \in PC_\infty$, we define

$$\|f\|_\infty := \operatorname*{ess\,sup}_{x \in [0, \infty)} \|f(x)\|.$$

2. THE PROBLEM STATEMENT

It is assumed that the SISO plant can be modeled by a linear time-invariant (LTI) system of the form

$$\dot{x} = Ax + Bu + Ew_1$$
$$y = Cx + Fw_2$$
$$e = y_{ref} - y\,,$$

where $x \in \mathbf{R}^n$ is the state, $u \in \mathbf{R}$ is the control input, $y \in \mathbf{R}$ is the plant output, $w_1 \in \mathbf{R}^{q_1}$ and $w_2 \in \mathbf{R}^{q_2}$ are unmeasurable disturbances, and $e \in \mathbf{R}$ is the error between the plant output y and the reference model output y_{ref}; we shall identify the plant with the 5-tuple (A, B, C, E, F). Since the actual reference model plays no role in the controllers to be presented, we shall dispense with the model and assume only that $y_{ref} \in PC^1_\infty$; this, of course, encompasses all possible outputs of a typical LTI reference model (which is stable and strictly proper) in response to a piecewise continuous and bounded input. Hence, we shall assume that only y and y_{ref} can be measured, and that only u can be excited.

The first objective is to design an adaptive controller, parametrized by a positive constant ε, such that for every $x(0) \in \mathbf{R}^n$, $y_{ref} \in PC^1_\infty$, $w_1 \in PC_\infty$, and $w_2 \in PC^1_\infty$, the closed loop system has the following properties:

(i) $|e(t)| < \varepsilon$ for all t sufficiently large, and

(ii) the plant state and the control signals are bounded on $t \geq 0$.

In other words, the control objective is for the closed system to exhibit "arbitrarily good" steady-state behaviour by choosing ε accordingly; hence we call this the *Arbitrarily Good Steady-State Response Problem (AGSSRP)*. Of course, we would like the adaptive controller to solve the AGSSRP for as large a class of plants as possible.

The second objective is to design an adaptive controller, parametrized by three positive constants ε, δ, and T, such that for every $x(0) \in \mathbf{R}^n$, $y_{ref} \in PC^1_\infty$, $w_1 \in PC_\infty$, and $w_2 \in PC^1_\infty$, the closed loop system has following properties:

(i) $(\mathrm{sgn}\, e(0))\, e(t) \in [-\varepsilon, \max\{\varepsilon, \delta + |e(0)|\}]$, $t \in [0, T]$,

(ii) $|e(t)| < \varepsilon$, $t \geq T$, and

(iii) the plant state and the control signals are bounded on $t \geq 0$;

the requirement on the error is shown graphically in Figure 1.

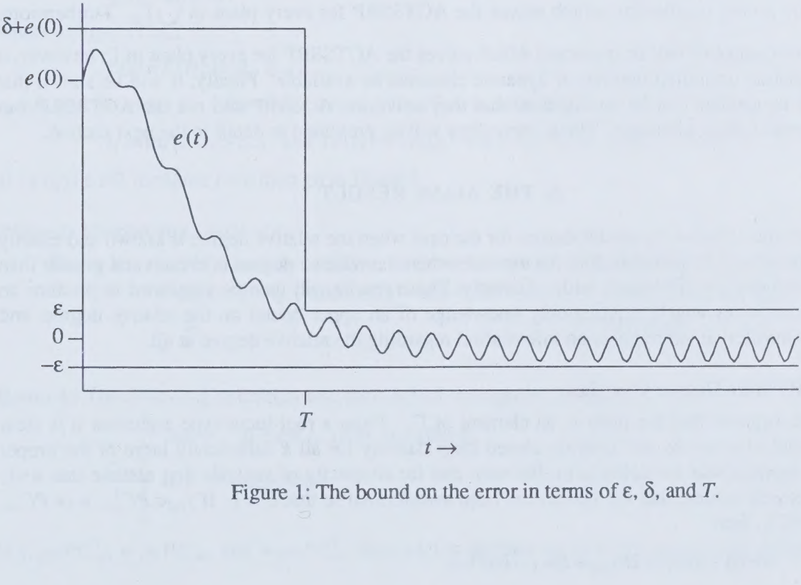


Figure 1: The bound on the error in terms of ε, δ, and T.

In other words, the control objective is that the closed loop system exhibit ''arbitrarily good'' transient and steady-state behaviour by choosing ε, δ, and T accordingly; hence, we call this the *Arbitrarily Good Transient and Steady-State Response Problem (AGTSSRP).* As in the AGSSRP, we would like the adaptive controller to solve the AGTSSRP for as large a class of plants as possible. Observe that any controller which solves the AGTSSRP automatically solves the AGSSRP, but not vice versa; we shall concentrate on controllers which solve both.

Throughout the paper it is assumed that (C, A, B) is minimum phase, i.e. that the polynomial

$$C \operatorname{adj}(sI-A)\, B = -\det \begin{bmatrix} A-sI & B \\ C & 0 \end{bmatrix}$$

has all of its zeros in $\mathbf{C}^-$; the zeros of this polynomial are said to be the zeros of (C, A, B). So if (C, A, B) is minimum phase, then the plant is stabilizable and detectable. A standing assumption is that $C \operatorname{adj}(sI-A)\, B$ is not the zero polynomial.

The relative degree p of the plant plays a crucial role here; it is the plant order n less the degree of $C \operatorname{adj}(sI-A)\, B$. Some *a priori* information about the relative degree will usually be required.

With q_1 and q_2 fixed but n arbitrary, for each $p \in \mathbf{N}$ define

$$\Gamma_p := \{(A,B,C,E,F) : (C,A,B) \text{ is minimum phase and has relative degree } p\},$$

and then define

$$\Gamma := \bigcup_{p=1}^{\infty} \Gamma_p .$$

For every $p \in \mathbf{N}$, an adaptive controller will be presented which solves the AGTSSRP for every plant in Γ_p; it consists of a $p-1^{th}$ order ''high-gain'' LTI compensator which is parametrized by a single parameter which is adjusted by a ''tuning mechanism'' at discrete points in time, i.e. it is a ''switching-type'' controller, e.g. [6], [7]. The tuning mechanism is itself parametrized by ε, δ, and T, and it relies on the fact that for a given $x(0) \in \mathbf{R}^n$, $y_{ref} \in PC^1_\infty$, $w_1 \in PC_\infty$, and $w_2 \in PC^1_\infty$, when the parameter being tuned is constant, the closed loop system is stable and the prescribed bound on the error is satisfied if this parameter is sufficiently large of the proper sign. For each $q \geq 2$, it will be shown that the LTI compensators for the $p = 2, 3, \ldots, q$ cases can be pieced together with an appropriate tuning mechanism which solves the AGTSSRP for every plant in $\bigcup_{i=1}^{q} \Gamma_p$. Furthermore, an adaptive controller will be presented which solves the AGTSSRP for every plant in Γ; however, it requires that an unlimited number of dynamic elements be available. Finally, it will be shown that the above controllers can be modified so that they solve the AGSSRP and not the AGTSSRP, but have improved noise tolerance. These controllers will be presented in detail in the next section.

3. THE MAIN RESULT

Since the adaptive controller design for the case when the relative degree is known and exactly one is somewhat different than that for the case when the relative degree is known and greater than one, the two cases will be dealt with separately. These results will then be combined to produce an adaptive controller which requires only knowledge of an upper bound on the relative degree, and another controller which requires no information regarding the relative degree at all.

3.1 The Relative Degree One Case

First, suppose that the plant is an element of Γ_1. From a root-locus type argument it is clear that the controller $u = ke$ will provide closed loop stability for all k sufficiently large of the proper sign. So suppose that we define u in this way, and for simplicity of analysis first assume that $n=1$; we may as well assume that the system has been transformed so that $C = 1$. If $y_{ref} \in PC^1_\infty$, $w_1 \in PC_\infty$, and $w_2 \in PC^1_\infty$, then

$$\dot{x} = (A - kB)x + kBy_{ref} + Ew_1 - kBFw_2$$

leads to

$$\begin{aligned}\dot e &= \dot y_{ref} - \dot y = \dot y_{ref} - (A-kB)x - kBy_{ref} - Ew_1 + kBFw_2 - F\dot w_2 \\ &= (A-kB)e - Ay_{ref} + \dot y_{ref} - Ew_1 + AFw_2 - F\dot w_2 .\end{aligned}$$

If $A-kB<0$, then

$$\begin{aligned}|e(t) - e^{(A-kB)t}e(0)| &\le \frac{1}{|A-kB|}\| -Ay_{ref} + \dot y_{ref} - Ew_1 + AFw_2 - F\dot w_2 \|_\infty \\ &\le \frac{\gamma_1}{|A-kB|},\ t \ge 0,\end{aligned}$$

where γ_1 is a suitably defined constant. Now suppose that $\varepsilon > 0$, $\delta > 0$, and $T > 0$. Then for k large and of the proper sign, it follows that $A-kB<0$, $\gamma_1/|A-kB| < \min\{\varepsilon, \delta\}/2$, and $|e^{(A-kB)T}e(0)| < \varepsilon/2$, so that $(sgn\, e(0))\, e(t) \in [-\varepsilon, \max\{\varepsilon, \delta + |e(0)|\}]$ for $t \in [0,T]$, $|e(t)| < \varepsilon$ for $t \ge T$, and $x(t)$ and $u(t)$ are bounded on $t \ge 0$. When the plant order is greater than one, it turns out that the closed loop system is basically a one-dimensional system with "parasitics" when k is "large", and the analysis of the closed loop system is similar to the above. Since we do not know the plant parameters nor the disturbances w_1 and w_2, we must somehow adjust the gain k as time proceeds.

We say that the function $f : \mathbf{N} \to \mathbf{R}$ is a tuning function of order 1 ($f \in TF_1$) if

$$\limsup_{i\to\infty} f(i)/2^i = \infty \quad \text{and} \quad \liminf_{i\to\infty} f(i)/2^i = -\infty,$$

e.g. if we define f by $f(i) = (-3)^i$, then $f \in TF_1$.

Now define the set of admissible controller parameters to be

$$\Omega_1 := \{(\varepsilon, \delta, T, f) : \varepsilon > 0, \delta > 0, T > 0, f \in TF_1\}.$$

With $\sigma = (\varepsilon, \delta, T, f) \in \Omega_1$, we define Controller 1 in two phases:

Phase 1: Force $|e(t)| \le \varepsilon/2$.

Set

$$u(t) = k(t)e(t)$$

where $k(0) = f(1)$,

$$k(t) = f(i),\ t \in (t_i, t_{i+1}],$$

$t_1 = 0$, and for every $i > 1$ for which $|e(t_{i-1})| > \varepsilon/2$, t_i is defined by

$$t_i := \inf\{t : t > t_{i-1}, \text{ and } |e(t)| = |e(0)| + \delta(1 - 1/2^i) \text{ or } |e(t)| \le \varepsilon/2 \text{ or } t = T(1 - 1/2^i)\}$$

If $|e(t_i)| \le \varepsilon/2$ for some $i = \bar i$, then go to Phase 2.

Phase 2: Maintaining $|e(t)| < \varepsilon$.

Define $u(t)$ and $k(t)$ exactly as above, but now for every $i > \bar i$ for which $t_{i-1} < \infty$, define t_i by

$$t_i := \begin{cases} \inf\{t : t > t_{i-1} \text{ and } |e(t)| = \varepsilon(1 - 1/2^{i-\bar i+1})\} & \text{if the set is nonempty} \\ \infty & \text{otherwise} \end{cases}$$

Remark: The following definition will be in effect throughout the paper:

$$\bar t := \begin{cases} \lim\limits_{i\to\infty} t_i & \text{if each } t_i,\ i \in \mathbf{N}, \text{ is defined} \\ \infty & \text{otherwise} \end{cases}$$

If $y_{ref} \in PC^1_\infty$, $w_1 \in PC_\infty$, and $w_2 \in PC^1_\infty$, then $x(t)$ is defined on $t \in [0, \bar t)$ and $e(t)$ is continuous on

$[0, \bar{t})$, so we can replace "inf" with "min" in the definition of t_i in both Phase 1 and Phase 2. Therefore, $(sgn\, e(0))\, e(t) \in [-\varepsilon, \max\ \{\varepsilon, \delta + |e(0)|\}]$ for $t \in [0, \min\{T, \bar{t}\})$, and $|e(t)| < \varepsilon$ for $t \in [T, \bar{t})$.

Remark: In Phase 1, we try to force $|e(t)| \le \varepsilon/2$. We switch gains if the error has become "too large" (in order to limit the transients), if we are getting "close" to $t = T$ but the error is still greater than $\varepsilon/2$ (in order to ensure that "approximate regulation" occurs quickly), or if $|e(t)| \le \varepsilon/2$ (so that we can switch to Phase 2). In Phase 2, we switch gains only if the error becomes "too large".

Theorem 1 [8]: Controller 1 solves the AGTSSRP for every plant in Γ_1.

Remark: In the course of the proof of Theorem 1, it is shown that eventually $k(t)$ stops changing, so we end up with a LTI closed loop system. This, in fact, is true of all of the controllers presented in this paper.

3.2 The Relative Degree $p > 1$ Case

Suppose that $p > 1$ and that the plant is an element of Γ_p. From a root locus type argument it follows that the controller $u = ke$ will not provide "high-gain stabilization" unless the plant belongs to a particular proper subset of Γ_2. So instead, consider the LTI controller

$$u = k^{2p-1} \frac{(s+1)^{p-1}}{(s+k^2)^{p-1}}\, e$$

which is shown in [9] to stabilize a minimum phase plant of relative degree p for all k sufficiently large in magnitude and of the proper sign. With α_i, $i = 0, 1, \ldots, p-1$, defined by

$$(s+1)^{p-1} = \sum_{i=0}^{p-1} \alpha_i s^i,$$

this controller can be implemented by

$$\dot{v} = k^2 G v + k^2 H e \tag{1a}$$

$$r = k J v \tag{1b}$$

$$u = \sum_{i=0}^{p-1} \alpha_i r^{(i)}, \tag{1c}$$

($r^{(i)}$ is defined to be the i^{th} derivative of r with respect to t) where

$$G = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 1 \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix} \in \mathbf{R}^{(p-1)\times(p-1)},\ H = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbf{R}^{p-1},\ J' = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbf{R}^{p-1},\ k \in \mathbf{R}.$$

It is easily seen that

$$v_1^{(i)} = k^{2i} \times \text{a linear combination of } e,\ v_1, \ldots, v_{p-1},\ i \in \mathbf{p-1}.$$

Since $r = kv_1$, it follows that u can be readily constructed from e and v.

Before presenting the self-tuning controller, define $f : \mathbf{N} \rightarrow \mathbf{R}$ to be a tuning function of order p ($f \in TF_p$) if

$$\limsup_{i \rightarrow \infty} \frac{f(i)}{2^i (1 + \max_{j \in \mathbf{i-1}} |f(j)|^{2p-1})} = \infty,$$

and

$$\liminf_{i\to\infty} \frac{f(i)}{2^i(1+\max_{j\in i-1} |f(j)|^{2p-1})} = -\infty .$$

For example, if we define f by $f(i)=(-i)^{i^i}$, $i\in \mathbf{N}$, then $f\in TF_p$ for all $p\in \mathbf{N}$.

Now define the set of admissible controller parameters to be

$$\Omega_p := \{(\varepsilon, \delta, T, f) : \varepsilon>0,\ \delta>0,\ T>0,\ f\in TF_p\}.$$

With $\sigma=(\varepsilon, \delta, T, f)\in\Omega_p$, we define Controller p in two phases:

Phase 1: Force $|e(t)|\le\varepsilon/2$.

Set

$$\dot v(t)=k(t)^2Gv(t)+k(t)^2He(t),\ v(t_i^+)=0,$$
$$r(t)=k(t)Jv(t),$$
$$u(t)=\sum_{j=0}^{p-1}\alpha_j r^{(j)}(t),$$

where $k(0)=f(1)$,

$$k(t)=f(i),\ t\in(t_i, t_{i+1}],$$

$t_1=0$, and for every $i>1$ for which $|e(t_{i-1})|>\varepsilon/2$, t_i is defined by

$$t_i := \inf\{t : t>t_{i-1}, \text{ and } |e(t)|=|e(0)|+\delta(1-1/2^i) \text{ or } |e(t)|\le\varepsilon/2 \text{ or } t=T(1-1/2^i)\}$$

If $|e(t_i)|\le\varepsilon/2$ for some $i=\bar i$, then go to Phase 2.

Phase 2: Maintaining $|e(t)|<\varepsilon$.

Define $u(t)$ and $k(t)$ exactly as above, but now for every $i>\bar i$ for which $t_{i-1}<\infty$, define t_i by

$$t_i := \begin{cases} \inf\{t : t>t_{i-1} \text{ and } |e(t)|=\varepsilon(1-1/2^{i-\bar i+1})\} & \text{if the set is nonempty} \\ \infty & \text{otherwise} \end{cases}$$

Remark: As for Controller 1, if $y_{ref}\in PC^1_\infty$, $w_1\in PC_\infty$, and $w_2\in PC^1_\infty$, then $e(t)$ is continuous for $t\in[0,\bar t)$ so we can replace "inf" with "min" in the definition of t_i in both phases. Therefore, $(sgn\, e(0))\, e(t)\in[-\varepsilon, \max\{\varepsilon, \delta+|e(0)|\}]$ for $t\in[0, \min\{T, \bar t\})$, and $|e(t)|<\varepsilon$ for $t\in[T, \bar t)$.

Theorem 2 [8]: For each $p>1$, Controller p solves the AGTSSRP for every plant in Γ_p.

3.3 The Exact Relative Degree Unknown

It would be desirable if one could replace the requirement that the relative degree be known with a weaker condition. Here we demonstrate that it suffices to know an upper bound on the relative degree, and in principle no knowledge regarding the relative degree is required.

It can be shown that Controller 2 solves the AGTSSRP for every plant in Γ_1 as well as every plant in Γ_2. Unfortunately, we cannot generalize this; in fact, if $p>3$ then the LTI controller

$$u=k^{2p-1}\frac{(s+1)^{p-1}}{(s+k^2)^{p-1}}e$$

will not provide "high-gain stabilization" for every minimum phase plant of relative degree less than or equal to p, e.g. see [10] (p. 94). However, it is shown in [10] (pp. 92-93) that the LTI controller

$$u=k^{2^p-1}\frac{(s+1)^{p-1}}{(s+k^2)(s+k^4)\cdots(s+k^{2^{p-1}})}e$$

does have this property, i.e. that for all k sufficiently large of the proper sign the closed loop system will be stable; in fact, an adaptive controller based on this structure is presented in [10] (pp. 92-95) and shown to provide stabilization when $y_{ref}=w_1=w_2=0$. Unfortunately, we have been unable to design an adaptive controller based on this LTI controller which will solve the AGTSSRP for every plant in this class.

Instead, we adopt a different approach. Suppose that it is known that the plant is in Γ and that the relative degree is less than or equal to $q>2$. Define $TF_q{}'$ by

$$TF_q{}' := \{f\in TF_q : f(i+1)f(i)\le 0,\ i\in \mathbf{N},\ \text{and} \lim_{i\to\infty} |f(i)| = \infty\} \subset \bigcap_{i=1}^{q} TF_i,$$

and

$$\Omega_q{}' := \{(\varepsilon,\delta,T,f) : \varepsilon>0,\ \delta>0,\ T>0,\ f\in TF_q{}'\};$$

so $f\in TF_q{}'$ means that the sign of $f(i)$ is the opposite of the sign of $f(i+1)$, at least for i sufficiently large so that $f(i)\ne 0$ and $f(i+1)\ne 0$. Now define $\rho:\mathbf{N}\to\mathbf{N}$ by

$$\rho(i) = 2 + (\text{integer part of } (i/2)) \text{ modulo } (q-1),\ i\in\mathbf{N}.$$

With $\sigma\in\Omega_q{}'$, define Controller q' to be basically the same as Controller q, but with the order of the LTI controller (1) equal to $\rho(i)-1$ for $t\in(t_i, t_{i+1}]$, i.e. the one which was used for plants with relative degree $\rho(i)$. So we switch between LTI controllers of order 1 through $q-1$, trying two different gains - one positive and one negative (for large i) - before switching orders once more; notice that we do not try a 0^{th} order LTI controller since it was shown above that the 1^{st} order controller works when the plant relative degree is one. Controller q' can in fact be viewed as a $q-1^{th}$ order LTI controller whose parameters are adjusted and whose state is set to zero at appropriate points in time.

Theorem 3 [8]: For each $q>2$, Controller q' solves the AGTSSRP for every plant in $\bigcup_{i=1}^{q}\Gamma_i$.

Now suppose that we do not even have an upper bound on the relative degree; in principle, we can do without it. Define Controller ∞ the same way that Controller q' was defined, except now we choose the controller parameter σ from

$$\Omega_\infty := \{(\varepsilon,\delta,T,f) : \varepsilon>0,\ \delta>0,\ T>0,\ f:\mathbf{N}\to R \text{ is defined by } f(i)=(-i)^{i^i}\},$$

and define ρ by

$$\rho(i) := \begin{cases} 2 & \text{if } i=1,2 \\ 3 & \text{if } i=3,4 \\ 2 & \text{if } i=5,6 \\ 3 & \text{if } i=7,8 \\ 4 & \text{if } i=9,10 \\ 2 & \text{if } i=11,12 \\ \vdots & \quad\vdots \end{cases}$$

So we "scan" the order of the LTI compensator, stopping to try each order for two different gains - one positive and one negative. Observe that

$$\Omega_\infty \subset \bigcap_{i=1}^{\infty} \Omega_i .$$

Theorem 4 [8]: Controller ∞ solves the AGTSSRP for every plant in Γ.

Remark: Although Controller ∞ does not require any information about the relative degree, it may be necessary to try LTI dynamic compensators of order much greater than the relative degree, and it is necessary that an unlimited number of dynamic elements be available from which to build these compensators, both of which are definite drawbacks.

3.4 Noise Tolerance

We have shown that each of the controllers presented thus far will solve the AGTSSRP (and therefore the AGSSRP as well) for a class of plants in an ideal, noise-free environment. Now consider the case when there is a noise term, which we shall denote by d, in the measurement of y. In the following discussion, let the controller be one of the controllers presented thus far, and assume that $y_{ref} \in PC_\infty^1$, $w_1 \in PC_\infty$, $w_2 \in PC_\infty^1$, and that the plant is in the set for which the controller has been shown to solve the AGTSSRP.

First, suppose that $d \in PC_\infty^1$, e.g. $d(t) = \sin \theta t$, $\theta \in \mathbf{R}$; this term will be viewed by the controller as part of y_{ref}, and the measured error $e_m(t) := y_{ref}(t) - y(t) - d(t)$ will be in the interval $[-\varepsilon, \max\{\varepsilon, \delta + |e(0)|\}]$ for $t \in [0, T]$ and less than ε in magnitude for $t \geq T$; so, in particular, the actual plant output will differ from y_{ref} by at most the sum of the magnitude of the noise term and ε for $t \geq T$.

Now suppose that d is bounded and continuous but not an element of PC_∞^1. Then the controller may not work; in fact, it may be that the gain "blows up" in a finite period of time. Also, in order to provide a good transient response, the gain may be switched very fast, especially during the initial T units of time. These two undesirable properties can be alleviated by modifying the controller; however, now only the AGSSRP is solved, and the transients may be quite large. This modification can be effected by combining Phase 1 and Phase 2 of the controller into a single phase and by redefining t_i by

$$t_i := \begin{cases} \inf\{t : t > t_{i-1} \text{ and } |e_m(t)| = \varepsilon(1 - 1/2^i) + (|e_m(0)| + i^2)e^{\lambda(t - t_{i-1})}\} & \text{if the set is nonempty} \\ \infty & \text{otherwise,} \end{cases}$$

where λ is an arbitrary negative constant, i.e. an envelope is put on $e_m(t)$, and if $e_m(t)$ hits this envelope then the gain is switched and the envelope changes. This modified controller solves the AGSSRP for the same class of plants as the original controller solves the AGTSSRP. Furthermore, if the noise term d is continuous and bounded on $[0, \infty)$, then the closed loop system is guaranteed to have a solution for all time; if in addition $|d(t)| < \varepsilon/2$ for large t, then $|e_m(t)| < \varepsilon$ for large t and all signals remain bounded. So if we know an upper bound on the size of the measurement noise in advance, then we can choose ε accordingly. Also, we would expect that in general the switching of the gain will not be as fast as in the original controller. This controller has effectively traded an arbitrarily good transient response for a better tolerance to sensor noise and perhaps a less frenzied gain switching.

4. CONCLUSIONS

In this paper we consider the problem of designing a controller which ensures that the error between the output of a partially known plant and the output of a reference model is less than an (arbitrarily small) prespecified constant after an (arbitrarily short) prespecified period of time, with an (arbitrarily small) prespecified upper bound on the amount of overshoot, for a large class of reference model input signals. To solve this problem, it is sufficient to assume that the plant is minimum phase, and that an upper bound on the relative degree is known; *knowledge of an upper bound on the plant order, of the exact relative degree, and of the sign of the high-frequency gain are not required.* This shows that if the exact regulation requirement in the standard model reference adaptive control problem is relaxed, then not only is the amount of information about the plant which is required to be known reduced considerably, but in addition we can design an adaptive controller which has an arbitrarily good transient response. The controller developed is shown to tolerate a large class of noise signals in the measurement of the plant output, and has dynamics of order equal to the upper bound on the relative degree less one.

Since each of the adaptive controllers presented here are based on tuning a "high-gain" LTI controller, the control signal may become very large, so saturation would play a crucial role in any attempt to practically implement such a controller. For this reason, the controllers should be viewed mainly as existence results, to show that in principle one can achieve an arbitrarily good transient and steady-state response with a relatively low order adaptive controller.

REFERENCES

[1] A.S. Morse, ''Global Stability of Parameter-Adaptive Control Systems'', *IEEE Transactions on Automatic Control,* vol. AC-25, pp. 433-439, June 1980.

[2] K.S. Narendra et. al., ''Stable Adaptive Controller Design, Part II: Proof of Stability'', *IEEE Transactions on Automatic Control,* vol. AC-25, pp. 440-448, June 1980.

[3] D.R. Mudgett and A.S. Morse, ''Adaptive Stabilization of Linear Systems with Unknown High-Frequency Gains'', *IEEE Transactions on Automatic Control,* vol. AC-30, pp. 549-554, June 1985.

[4] A.S. Morse, ''A 4(n+1)-dimensional Model Reference Adaptive Stabilizer for any Relative Degree One or Two, Minimum Phase System of Dimension n or Less'', *Automatica,* vol. 23, No. 1, pp. 123-125, 1987.

[5] K.S. Narendra and A.M. Annaswamy, *Stable Adaptive Systems,* to be published by Prentice Hall, Inc., New Jersey, in 1988.

[6] M. Fu and B.R. Barmish, ''Adaptive Stabilization of Linear Systems Via Switching Control'', *IEEE Transactions on Automatic Control,* vol. AC-31, pp. 1097-1103, Dec. 1986.

[7] D.E. Miller and E.J. Davison, ''A New Self-Tuning Controller to Solve the Servomechanism Problem'', *Proc. IEEE 26th Conf. on Decision and Control,* Dec. 1987.

[8] D.E. Miller and E.J. Davison, ''An Adaptive Controller to Achieve An Arbitrarily Good Transient And Steady-State Response'', *Systems Control Report No. 8808,* Dept. of Electrical Engineering, University of Toronto, in preparation.

[9] H.K. Khalil and A. Saberi, ''Adaptive Stabilization of a Class of Nonlinear Systems Using High-Gain Feedback'', *IEEE Transactions on Automatic Control,* vol. AC-32, pp.1031-1035, Nov. 1987.

[10] B. Martensson, *Doctoral Dissertation,* Dept. of Aut. Control, Lund Institute of Technology, Sweden, 1986.

A Robust Adaptive Control For Linear Plants in The Presence of Unmodeled Dynamics

Shi Bai and Fathi M. A. Salam

Department of Electrical Engineering and Systems Science,
Michigan State University, East Lansing, MI 48824

ABSTRACT

We present a robust direct adaptive control algorithm with respect to additive and/or multiplicative plant unmodeled dynamics. The plant may be nonminimum phase and of unknown order and of unknown relative degree. The transfer function of the modeled part of the plant however is assumed to be minimum phase and of known order and of known relative degree. We give a complete stability and convergence proofs for such a class of adaptive control systems. In the absence of unmodeled dynamics, this algorithm guarantees the zero tracking error. Also when the reference input is a constant, zero tracking error is ensured even in the face of unmodeled dynamics. For a time varying reference input and in the presence of unmodeled dynamics, the error converges to a sufficiently small set.

Keywords: Robustness, Adaptive Control system, Unmodeled dynamics.

1. Introduction

Robustness of an adaptive system with respect to certain perturbations and unmodeled dynamics has become an important measure for performance of the overall system under such uncertainties. For plants with unmodeled dynamics, several methods and algorithms have been developed [1]-[4]. The dead-zone approach, and its modifications, have been used in the stability and convergence analysis. Those analyses show that by using these modification schemes, it is ensured that, when there are unmodeled dynamics, there exists a bounded region and all signals will converge to this bounded region. But in many cases, especially in the n–dimensional space, the estimation or determination of the boundary of the convergence region often shows that the size of the region is quite large.

In this paper we analyze an adaptive control system in which the plant has the additive and/or multiplicative unmodeled dynamics. We give a complete stability and convergence proofs of the adaptive control algorithm presented here. Our analysis shows that, in the absence of unmodeled dynamics, this algorithm guarantees zero tracking errors. By using this algorithm for a constant reference input the tracking errors will asymptotically converge to zero independent of the presence of unmodeled dynamics. A sufficiently small tracking errors are ensured for time varying reference inputs when unmodeled dynamics are present. The error bound in that case can easily be estimated. The transfer function of the modeled part of the plant is assumed to be minimum phase and of known order and of known relative degree. The actual plant however can be nonminimum phase and of unknown order and relative degree.

The presentation is organized as follows. In section 2, we introduce the model of the plant with the additive and multiplicative unmodeled dynamics. we give the assumptions on the modeled part of the plant transfer function and the unmodeled dynamics. In section 3, we formulate the system and give a complete stability and convergence proofs. In section 4, we list some conclusions.

2. An Adaptive Control System

Given the transfer function of a plant (SISO)

$$\frac{Y(s)}{U(s)} = G(s) = G_0(s)[\ 1 + \mu_2\Delta_2(s)\ ] + \mu_1\Delta_1(s) \tag{1}$$

where

$G(s)$: strictly proper transfer function of the plant

$G_0(s)$: minimal phase, strictly proper transfer function of the modeled part of the plant

$\mu_1\Delta_1(s)$: additive plant unmodeled dynamics

$\mu_2\Delta_2(s)$: multiplicative plant unmodeled dynamics

$\Delta_1(s)$: strictly proper stable transfer function

$\Delta_2(s)$: stable transfer function.

Rewrite (1) as

$$\begin{aligned} Y(s) &= G_0(s)U(s) + \mu_2 G_0(s)\Delta_2(s)U(s) + \mu_1\Delta_1(s)U(s) \\ &= G_0(s)U(s) + \mu_2\overline{\Delta}_2(s)U(s) + \mu_1\overline{\Delta}_1(s)U(s) \end{aligned} \tag{2}$$

where

$$\overline{\Delta}_1(s) = \Delta_1(s),$$

$$\overline{\Delta}_2(s) = G_0(s)\Delta_2(s).$$

We also assume that we know a lower bound $p_1 > 0$ and a lower bound $p_2 > 0$ on the stability margin for which the poles of $\overline{\Delta}_1(s - p_1)$ and $\overline{\Delta}_2(s - p_2)$ are stable.

Define

$$\begin{aligned} Y_0(s) &= G_0(s)U(s), \\ Y_1(s) &= \overline{\Delta}_1(s)U(s), \\ Y_2(s) &= \overline{\Delta}_2(s)U(s). \end{aligned} \tag{3}$$

Now, let $(A_0\ ,\ b_0\ ,\ h_0^T)$ be a minimal realization of $G_0(s)$,

$(A_1\ ,\ b_1\ ,\ h_1^T)$ be a minimal realization of $\overline{\Delta}_1(s)$,

$(A_2\ ,\ b_2\ ,\ h_2^T)$ be a minimal realization of $\overline{\Delta}_2(s)$,

then (2) becomes

$$Y(s) = G(s)U(s) = Y_0(s)+\mu_1 Y_1(s) + \mu_2 Y_2(s). \tag{4}$$

So, a realization of $G(s)$ is

$$\begin{aligned} \dot{x}_0 &= A_0 x_0 + b_0 u \\ y &= h_0^T x_0 + \mu_1 h_1^T x_1 + \mu_2 h_2^T x_2 \end{aligned} \tag{5i}$$

$$\begin{aligned} \dot{x}_1 &= A_1 x_1 + b_1 u \quad , \quad & \dot{x}_2 &= A_2 x_2 + b_2 u \\ y_1 &= h_1^T x_1 \quad , & y_2 &= h_2^T x_2 \end{aligned} \tag{5ii}$$

The reference model is

$$\begin{aligned} \dot{x}_m &= A_m x_m + b_m r \\ y_m &= h_m^T x_m \end{aligned} \tag{6}$$

where r is reference input.

The plant transfer function $G(s)$ may be nonminimum phase even when $G_0(s)$ is assumed to be minimum phase.

3. Stability and Convergence

we use the input u and output y to generate a $(2n - 1)$–dimensional auxiliary vector as

$$\dot{\omega}_1 = F\omega_1 + qu\ ,\ \dot{\omega}_2 = F\omega_2 + qy \tag{7}$$

where F is a stable matrix.

Define

$$\omega^T = [r, \omega_1^T, y, \omega_2^T]\ ,\ \theta^T = [c_0, c^T, d_0, d^T]$$

then

$$u = \theta^T \omega$$

Now we have

$$\dot{X}_c = \begin{bmatrix} \dot{x}_0 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} A_0 & 0 & 0 \\ 0 & F & 0 \\ qh_0{}^T & 0 & F \end{bmatrix} \begin{bmatrix} x_0 \\ \omega_1 \\ \omega_2 \end{bmatrix} + \begin{bmatrix} b_0 \\ q \\ 0 \end{bmatrix} u + \mu_1 \begin{bmatrix} 0 \\ 0 \\ q \end{bmatrix} h_1{}^T x_1 + \mu_2 \begin{bmatrix} 0 \\ 0 \\ q \end{bmatrix} h_2{}^T x_2. \tag{8}$$

Define $\phi = \theta - \theta^*$, thus the control input can be expressed as

$$\begin{aligned} u = \theta^T \omega = \phi^T \omega + c_0^* r + c^{*T} \omega_1 + d_0^* h_0^T x_0 \\ + \mu_1 d_0^* h_1^T x_1 + \mu_2 d_0^* h_2^T x_2 + d^{*T} \omega_2 \end{aligned} \tag{9}$$

and, rewrite equation (8) as

$$\begin{aligned} \dot{X}_c = \begin{bmatrix} A_0 + d_0^* b_0 h_0^T & b_0 c^{*T} & b_0 d^{*T} \\ d_0^* q h_0^T & F + q c^{*T} & q d^{*T} \\ q h_0^T & 0 & F \end{bmatrix} X_c + \begin{bmatrix} b_0 \\ q \\ 0 \end{bmatrix} (\phi^T \omega + c_0^* r) \\ + \mu_1 \begin{bmatrix} d_0^* b_0 \\ d_0^* q \\ q \end{bmatrix} h_1^T x_1 + \mu_2 \begin{bmatrix} d_0^* b_0 \\ d_0^* q \\ q \end{bmatrix} h_2^T x_2. \end{aligned} \tag{10}$$

So,

$$\dot{X}_c = A_c X_c + b_c(\phi^T \omega + c_0^* r) + \mu_1 b_{c1} h_1^T x_1 + \mu_2 b_{c2} h_2^T x_2 \tag{11}$$

$$y = h_c^T X_c + \mu_1 h_1^T x_1 + \mu_2 h_2^T x_2 \tag{12}$$

and the augmented reference model from (6) is given by

$$\dot{X}_{mc} = A_c X_{mc} + b_c c_0^* r \tag{13}$$

$$y_m = h_c^T X_{mc}. \tag{14}$$

Define

$$e = X_c - X_{mc}, \quad e_1 = y - y_m. \tag{15}$$

Then,

$$\dot{e} = A_c e + b_c \phi^T \omega + \mu_1 b_{c1} h_1^T x_1 + \mu_2 b_{c2} h_2^T x_2 \tag{16}$$

$$e_1 = h_c e + \mu_1 h_1^T x_1 + \mu_2 h_2^T x_2 \tag{17}$$

We also have

$$\dot{x}_1 = A_1 x_1 + \mu_1 d_0^* b_1 h_1^T x_1 + \mu_2 d_0^* b_1 h_2^T x_2 + b_1(\phi^T \omega + c_0^* r) + b_1 \nu^T e + b_1 \nu^T X_{mc} \tag{18}$$

$$\dot{x}_2 = A_2 x_2 + \mu_1 d_0^* b_2 h_1^T x_1 + \mu_2 d_0^* b_2 h_2^T x_2 + b_2(\phi^T \omega + c_0^* r) + b_2 \nu^T e + b_2 \nu^T X_{mc} \tag{19}$$

$$\dot{X}_{mc} = A_c X_{mc} + b_c c_0^* r \tag{20}$$

where $\nu^T = [d_0^* h^T \, , c^{*T} \, , d^{*T}]$

Let us define:

$$\bar{A}_1 = A_1 + \mu_1 d_0^* b_1 h_1^T, \qquad \bar{A}_2 = A_2 + \mu_2 d_0^* b_2 h_2^T$$

Thus,

$$\dot{e} = A_c e + b_c \phi^T \omega + \mu_1 b_{c1} h_1^T x_1 + \mu_2 b_{c2} h_2^T x_2 \tag{21}$$

$$e_1 = h_c e + \mu_1 h_1^T x_1 + \mu_2 h_2^T x_2 \tag{22}$$

$$\dot{x}_1 = A_1 x_1 + \mu_1 d_0^* b_1 h_1^T x_1 + \mu_2 d_0^* b_1 h_2^T x_2 + b_1(\phi^T \omega + c_0^* r) + b_1 \nu^T e + b_1 \nu^T X_{mc} \tag{23}$$

$$\dot{x}_2 = A_2 x_2 + \mu_1 d_0^* b_2 h_1^T x_1 + \mu_2 d_0^* b_2 h_2^T x_2 + b_2(\phi^T \omega + c_0^* r) + b_2 \nu^T e + b_2 \nu^T X_{mc} \tag{24}$$

$$\dot{X}_{mc} = A_c X_{mc} + b_c c_0^* r \tag{25}$$

Now we assume that (It will be proved later)

$$H_c(s) = h_c^T [sI - A_c]^{-1} b_c, \tag{26}$$

$$H_1(s) = h_1^T [sI - \bar{A}_1]^{-1} b_1, \tag{27}$$

$$H_2(s) = h_2^T [sI - \bar{A}_2]^{-1} b_2 \tag{28}$$

are strictly positive real.
Thus, there exist P_c , P_1 , P_2 , Q_c , Q_1 , $Q_2 > 0$ such that

$$A_c^T P_c + P_c A_c = -Q_c \, , \qquad P_c b_c = h_c, \tag{29}$$

$$\bar{A}_1^T P_1 + P_1 \bar{A}_1 = -Q_1 \, , \qquad P_1 b_1 = h_1, \tag{30}$$

$$\bar{A}_2^T P_2 + P_2 \bar{A}_2 = -Q_2 \, , \qquad P_2 b_2 = h_2. \tag{31}$$

Define

$$\dot{\phi} = -\Gamma e_1 \omega - \Gamma \gamma \theta, \quad \gamma = \begin{cases} 0, & \|\theta\| < M \\ \gamma(1 - e^{M - \|\theta\|^2}), & M \leq \|\theta\| \end{cases} \tag{32}$$

where $\Gamma = \Gamma^T > 0$ and M is an upper bound of $\|\theta^*\|$, and $\gamma > 0$.

Remark 3.1: We assume that we have the knowledge of an upper bound M on the norm of the desired controller parameter vector $\|\theta^*\|$.

Choose (assume that $\mu_1 \geq 0$, $\mu_2 \geq 0$)

$$\begin{aligned} V(e \, , x_1 \, , x_2 \, , \phi \, , X_{mc}) = {} & e^T P_c e + \mu_1 x_1^T P_1 x_1 + \mu_2 x_2^T P_2 x_2 + \phi^T \Gamma^{-1} \phi \\ & + \alpha X_{mc}^T P_c X_{mc} \end{aligned} \tag{33}$$

where $\alpha > 0$.
Then,

$$\dot{V}(e \, , x_1 \, , x_2 \, , \phi \, , X_{mc}) = -e^T Q_c e + 2\mu_1 e^T P_c b_{c1} h_1^T x_1 + 2\mu_2 e^T P_c b_{c2} h_2^T x_2$$

$$\begin{aligned}
&- \mu_1 x_1^T Q_1 x_1 + 2\mu_1\mu_2 d_0^* x_1^T h_1 h_2^T x_2 + 2\mu_1 c_0^* x_1^T h_1 r \\
&+ 2\mu_1 x_1^T h_1 \nu^T e + 2\mu_1 x_1^T h_1 \nu^T X_{mc} \\
&- \mu_2 x_2^T Q_2 x_2 + 2\mu_1\mu_2 d_0^* x_2^T h_2 h_1^T x_1 + 2\mu_2 x_2^T h_2 c_0^* r \\
&+ 2\mu_2 x_2^T h_2 \nu^T e + 2\mu_2 x_2^T h_2 \nu^T X_{mc} - 2\gamma \phi^T \Gamma^{-1} \theta. \\
&- \alpha X_{mc}^T Q_c X_{mc} + 2\alpha X_{mc}^T P_c b_c c_0{}^* r
\end{aligned} \tag{34}$$

Let

$$\begin{aligned}
&3\lambda_c = \min\{\lambda(Q_c)\}, \quad 7\lambda_1 = \min\{\lambda(Q_1)\}, \quad 7\lambda_2 = \min\{\lambda(Q_2)\}, \\
&M_c = \max\{\lambda(P_c)\}, \quad M_1 = \max\{\lambda(P_1)\}, \quad M_2 = \max\{\lambda(P_2)\}, \\
&M_0 = \max\{\lambda(\Gamma^{-1})\}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\dot{V}(e\,,x_1\,,x_2\,,\phi\,,X_{mc}) \le\ & - \lambda_c \|e\|^2 - \mu_1\lambda_1\|x_1\|^2 - \mu_2\lambda_2\|x_2\|^2 - 2\gamma M_0\|\phi\|^2 \\
& - (\lambda_c - \mu_1 \frac{M_c^2}{\lambda_1}\|b_{c1}\|^2\|h_1\|^2 - \frac{\mu_1}{\lambda_1}\|h_1\|^2\|\nu\|^2)\|e\|^2 \\
& - (\lambda_c - \mu_2 \frac{M_c^2}{\lambda_2}\|b_{c2}\|^2\|h_2\|^2 - \frac{\mu_2}{\lambda_2}\|h_2\|^2\|\nu\|^2)\|e\|^2 \\
& - \mu_1(\lambda_1 - \frac{\mu_1\mu_2}{\lambda_2} d_0{}^{*2}\|h_1\|^2\|h_2\|^2)\|x_1\|^2 \\
& - \mu_2(\lambda_2 - \frac{\mu_1\mu_2}{\lambda_1} d_0{}^{*2}\|h_1\|^2\|h_2\|^2)\|x_2\|^2 \\
& - [\alpha\lambda_c - (\frac{\mu_1}{\lambda_1}\|h_1\|^2 + \frac{\mu_2}{\lambda_2}\|h_2\|^2)\|\nu\|^2\|]X_{mc}\|^2 \\
& - 2\alpha\lambda_c(\|X_{mc}\| - \frac{\|h_c\||c_0{}^* r|}{2\lambda_c})^2 \\
& + [(\frac{\mu_1}{\lambda_1}\|h_1\|^2 + \frac{\mu_2}{\lambda_2}\|h_2\|^2) + \alpha\frac{\|h_c\|^2}{2\lambda_c}]c_0{}^{*2} r^2.
\end{aligned} \tag{35}$$

Now, we want that

(i) $\lambda_c - \mu_1 \frac{M_c^2}{\lambda_1}\|b_{c1}\|^2\|h_1\|^2 - \frac{\mu_1}{\lambda_1}\|h_1\|^2\|\nu\|^2 \ge 0$

that is

$$\mu_1 \le \frac{\lambda_1\lambda_c}{(M_c^2\|b_{c1}\|^2 + \|\nu\|^2)\|h_1\|^2} \tag{36}$$

(ii) $\lambda_c - \mu_2 \frac{M_c^2}{\lambda_2} \|b_{c2}\|^2 \|h_2\|^2 - \frac{\mu_2}{\lambda_2} \|h_2\|^2 \|\nu\|^2 \geq 0$

that is

$$\mu_2 \leq \frac{\lambda_2 \lambda_c}{(M_c^2 \|b_{c2}\|^2 + \|\nu\|^2)\|h_2\|^2} \tag{37}$$

(iii) $\lambda_1 - \frac{\mu_1 \mu_2}{\lambda_2} d_0{*}^2 \|h_1\|^2 \|h_2\|^2 \geq 0$

that is

$$\mu_1 \mu_2 \leq \frac{\lambda_1 \lambda_2}{d_0{*}^2 \|h_1\|^2 \|h_2\|^2}. \tag{38}$$

(iiii) $\lambda_c - \frac{1}{\alpha}\left(\frac{\nu_1}{\lambda_1}\|h_1\|^2 + \frac{\nu_2}{\lambda_2}\|h_2\|^2\right)\|\nu\|^2 > 0$

The conditions (iiii) can be easily satisfied by properly choosing the values of λ_c and α.

Let

$$\mu_1{*} = \min\left\{ \frac{\lambda_1 \lambda_c}{(M_c^2 \|b_{c1}\|^2 + \|\nu\|^2)\|h_1\|^2} , \frac{\lambda_1 (M_c^2 \|b_{c2}\|^2 + \|\nu\|^2)}{d_0{*}^2 \lambda_c \|h_1\|^2} \right\} \tag{39}$$

$$\mu_2{*} = \min\left\{ \frac{\lambda_2 \lambda_c}{(M_c^2 \|b_{c2}\|^2 + \|\nu\|^2)\|h_2\|^2} , \frac{\lambda_2 (M_c^2 \|b_{c1}\|^2 + \|\nu\|^2)}{d_0{*}^2 \lambda_c \|h_2\|^2} \right\} \tag{40}$$

Thus, for $\mu_1 \in [0 , \mu_1{*}]$, $\mu_2 \in [0 , \mu_2{*}]$, we have

$$\dot{V}(e , x_1 , x_2 , \phi , X_{mc}) \leq - \delta V(e , x_1 , x_2 , \phi , X_{mc}) - 2\alpha\lambda_c \left(\|X_{mc}\| - \frac{\|h_c\| |c_0{*} r|}{2\lambda_c}\right)^2$$
$$+ \left[\left(\frac{\mu_1}{\lambda_1}\|h_1\|^2 + \frac{\mu_2}{\lambda_2}\|h_2\|^2\right) + \alpha \frac{\|h_c\|^2}{2\lambda_c}\right] c_0{*}^2 r^2. \tag{41}$$

where

$$\delta = \min\left\{ \frac{\lambda_c}{M_c} , \frac{\lambda_1}{M_1} , \frac{\lambda_2}{M_2} , \frac{2\gamma}{M_0} \right\}$$

When the reference input r is a constant, we can choose $X_{mc}(0) = A_c^{-1} b_c c_0{*} r$, then

$X_{mc}(t) = X_{mc}(0)$ for all t. Now if we choose

$$\alpha = \frac{(\|A_c^{-1}b_c\|^2\|v\|^2 + 1)(\frac{\mu_1}{\lambda_1}\|h_1\|^2 + \frac{\mu_2}{\lambda_2}\|h_2\|^2)}{3\lambda_c\|A_c^{-1}b_c\|^2 + 2\|A_c^{-1}b_c\|\|h_c\|},$$

then, the tracking error converges to zero when the reference input is a constant and/or the unmodeled dynamics are absent.

Now we are going to prove the assumption of strictly positive real (SPR) of H_c, H_1 and H_2 [6].

1) Since $W_m(s)$ is SPR, and $H_c(s) = W_m(s)$. Then, $H_c(s)$ is also SPR.

2) For proving $H_1(s)$ is SPR, we need to show that $H_1(s)$ is stablizable and completely controllable.
First, we assume that $\tilde{A}_1$ is stable (it will be proved later.). Then, $H_1(s)$ is stablizable.
Second, we want to show that $(\tilde{A}_1 , b_1)$ is completely controllable.

$$\tilde{A}_1 = A_1 + \mu_1 b_1 d_0{}^* h_1^T , \qquad b_1 = b_1. \tag{43}$$

We know that (A_1 , b_1) is completely controllable and

$$\tilde{A}_1 = A_1 + \mu_1 b_1 d_0{}^* h_1^T = A_1 + b_1 k_1 , \qquad k_1 = \mu_1 d_0{}^* h_1^T. \tag{44}$$

Then, $(\tilde{A} , b_1)$ is also completely controllable [7]. So $H_1(s)$ is SPR.

3) For proving $H_2(s)$ is SPR, by assuming that $\tilde{A}_2$ is stable and going through the similar procedures as in 2), we can show that $H_2(s)$ is also SPR.
Now, we are going to prove that $\tilde{A}_1$ and $\tilde{A}_2$ are stable.
rewrite

$$\tilde{A}_1 = A_1 + \mu_1 b_1 d_0{}^* h_1^T$$

For ease of presentation, we assume that A_1 is in the diagonal form.

The eigenvalues of $\tilde{A}_1 =$ roots of $Det(sI - \tilde{A}_1)$

$$Det(sI - \tilde{A}_1) = Det[sI - (A_1 + \mu_1 d_0{}^* b_1 h_1^T)] \tag{45}$$

Let

$$\hat{A}_1 = \mu_1 d_0{}^* h_1^T, \quad [\hat{A}_1]_{ij} = \hat{a}_{ij}$$

and $\lambda_i^1 = i^{th}$ eigenvalue of A_1. Then, by

Gershgorin Theorem [5]:

The union of all disc

$$K_i = \left\{ \gamma \in \mathbf{C} \middle| \, |\gamma - \lambda_i^1 - \hat{a}_{ii}| \le \sum_{\substack{j=1 \\ j \ne i}}^{n_1} |\hat{a}_{ij}| \right\} \tag{46}$$

contains all eigenvalues of $\tilde{A}_1$.

So, using the above theorem, the eigenvalues of $\tilde{A}_1$ are in the open LHP if

$$|\lambda_i^1 + \hat{a}_{ii}| > \sum_{\substack{j=1 \\ j \ne i}}^{n_1} |\hat{a}_{ij}|$$

=>

$$|\lambda_i^1| > |\hat{a}_{ii}| + \sum_{\substack{j=1 \\ j \ne i}}^{n_1} |\hat{a}_{ij}|$$

=>

$$|\lambda_i^1| > \sum_{j=1}^{n_1} |\hat{a}_{ij}| \tag{47}$$

Since A_1 is stable, then $\lambda_i^1 < 0$. Then $\tilde{A}_1$ is stable if

$$\lambda_i^1 < -\mu_1 |d_0{}^*| \|b_1\| \|h_1^T\| \tag{48}$$

=>

$$\lambda_i^1 < -\max \left\{ \frac{\lambda_1 \lambda_c |d_0{}^*| \|b_1\|}{(M_c^2 \|b_{c1}\|^2 + \|\nu\|^2) \|h_1\|}, \frac{\lambda_1 (M_c^2 \|b_{c2}\|^2 + \|\nu\|^2) \|h_1\|}{\lambda_c |d_0{}^*|} \right\}. \tag{49}$$

For proving $\tilde{A}_2$ being stable, we go through the similar procedures as in proving $\tilde{A}_1$ being stable, and we have that $\tilde{A}_2$ is stable if

$$\lambda_i^2 < -\max \left\{ \frac{\lambda_2 \lambda_c |d_0{}^*| \|b_2\|}{(M_c^2 \|b_{c2}\|^2 + \|\nu\|^2) \|h_2\|}, \frac{\lambda_2 (M_c^2 \|b_{c1}\|^2 + \|\nu\|^2) \|h_2\|}{\lambda_c |d_0{}^*|} \right\}. \tag{50}$$

We can always find the proper values of λ_1 , λ_2 , M_c such that (39), (40), (49) and (50) are satisfied.

4. Conclusions

(1) A complete stability and convergence proofs are given for an adaptive control system where the plant has additive and multiplicative unmodeled dynamics.

(2) All signals in the overall system are bounded. In the absence of unmodeled dynamics, this algorithm guarantees zero tracking errors.

(3) For a constant reference input, irrespective of the presence of unmodeled dynamics, the tracking error asymptotically converges to zero.

(4) A sufficiently small tracking error may exist when there are unmodeled dynamics and the reference input is varying with time.

(5) The estimation of the error bound becomes much simpler in this case.

5. References

[1] P. A. Ioannou and K. S. Tsakalis, "A robust direct adaptive controller," *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 11, pp. 1033-1043, 1986

[2] B. B. Peterson and K. S. Narendra, "Bounded error adaptive control," *IEEE Trans. Automat. Contr.*, vol. AC-27, no. 6, pp. 1161-1168, 1982.

[3] P. A. Ioannou and P. V. Kokotovic, "Robust redesign of adaptive control," *IEEE Trans. Automat. Contr.*, vol. AC-29, no. 3, Mar. 1984.

[4] K. S. Narendra and A. M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 134-145, 1987.

[5] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. New York, Springer-Verlag

[6] Anderson and Vongranitlerd, *Network Analysis and Synthesis,* Prentice Hall, 1973.

[7] C. T. Chen, *Linear System Theory and Design,* HRW, 1984.

LIAPUNOV BASED MODEL REFERENCE LEARNING CONTROL[1]

MINH PHAN[2] and RICHARD W. LONGMAN[3]
Columbia University
New York, New York 10027

ABSTRACT

When repetitive processes involve the use of controllers, the same controller error in following the command is repeated with every repetition of the process. Here a learning controller is generated which modifies the control for each repetition based on previous experience in performing the task, in order to eliminate the tracking error. Unlike previous learning control results, the learning control law is guaranteed to produce a stable learning process for substantial classes of control problems, in spite of lack of knowledge about the system. The results represent a learning control analog to Liapunov based model reference adaptive control.

INTRODUCTION

Only in special circumstances will a control system give a perfect response to a command. This discrepency between response and desired response can be particularly important in tracking problems where the input can be constantly changing. Some understanding of the nature of this error can be obtained by considering a control system whose transfer function has no zeros and has all real roots to its characteristic equation. Then the zero initial condition particular solution of the system differential equation can be interpreted as a weighted average of past commands, or as a repeated weighted average, with the weighting factors emphasizing recent commands. Thus, the particular solution response is an averaged version of recent commands, and is therefore a somewhat smoothed or somewhat delayed version of the requested response. To this error one needs to add the transient response for the actual intial conditions.

When tracking control systems are employed in repetitive operations, for example in the use of robots in various manufacturing processes, it seems a bit primitive to let the controllers make the same tracking errors over and over again. This paper develops a learning control strategy, in which the usual feedback control signal is modified in each repetition of the repetitive task based on previous experience in performing the task.

When the command is something which the system is capable of performing, the learning control can learn to eliminate the transient error and the smoothing or delay like errors discussed above. It can do more. It is common that a control system performing a repetitive task will be subject to a repetitive disturbance, and the learning control can eliminate errors from this source as well. For example, in repetitive robot tracking, the centrifugal effects produced by motion of inner links are treated as disturbances to the control systems of outer links. In a repetitive operation, these effects become repetitive disturbances, whose effects can be eliminated by learning control.

Early works relating to learning control can be found in References [1-3], and more recent works appear in [4-25]. The use of the term "learning control" is not entirely standardized in the literature, and the

---

[1] Research supported by NASA Grant NAG 1-649.

[2] Participant, NASA Graduate Researcher Program, NASA Langley Research Center.

[3] Professor of Mechanical Engineering.

phrase "betterment process" is sometimes used for the type of control considered here. Reference [25] develops a formulation of learning control which we make heavy use of here.

THE BASIC OBJECTIVE

The literature suggests various learning control laws which will cause the system trajectory to converge to the desired trajectory -- provided the learning operator creates a stable learning process. Unfortunately, to date the stability of all such learning processes depends on the system equations as well as the learning operator gains. There is a fundamental dilemma here: only if one knows the system accurately, can one insure that the learning process will converge, but if one knows the system, there is no need to learn, since one can design the controller to give the desired trajectory directly.

The basic idea of this paper is to develop a learning control law based on Liapunov stability concepts which is guaranteed for a substantial class of systems to produce the desired trajectory -- in spite of lack of knowledge of the system matrices. We succeed in converting adaptive control concepts to the learning control problem, which differs in that it has both a time history and a repetition history. With [26,27] as the starting point the nontrivial extension to discrete systems was generated, and from this a direct model reference learning control law is obtained. In a companion work [28], we develop different learning control algorithms reminiscent of indirect adaptive control, again with guaranteed stability.

The guaranteed convergence of these algorithms represents a very important improvement over existing learning control methods.

THE SYSTEM

The system to be controlled is of the form

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \qquad (1)$$
$$k = 0,1,2,\dots,p-1$$

This p-step process is repeated many times, starting from the same initial condition $x(0)$. As indicated, the system can be time-varying. The $u(k) \in R^m$ is the learning control signal which will be chosen for each repetition based on the system performance during previous experience at the same task. If the process contains a feedback control, it has already been substituted into the equation so that A is the system matrix of the closed loop system. The actual control applied during any repetition is then the sum of the feedback control signal and the learning control u obtained here. The $w(k)$ represents any repetitive disturbance that might be present. The state vector $x(k) \in R^{n_s}$, where n_s is known, and the choice of state variables is made such that each of the measurements received at any time step are measurements of some component of x. The matrices A and B are assumed to be unknown.

SPECIFICATION OF A REFERENCE MODEL WITH LEARNING

The Model

The approach used to generate a learning controller with guaranteed convergence makes use of a reference model. A learning controller is used with the reference model, and since the model is known, it is easy to obtain a learning controller with guaranteed convergence by the methods of [25]. Then the output of the reference model converges to the desired output as the repetitions progress. A Liapunov based learning control is designed to insure that the system output converges to the model reference output, and hence it also converges to the desired output. The rate of convergence for the reference model can be made arbitrarily fast in the sense that the error can have a deadbeat response. The Liapunov based learning control will also have parameters that can be adjusted to influence convergence. Reference [25] discusses and quantifies the tradeoff in learning speed -- learning too fast increases the sensitivity of the system response to noise.

The reference model is

$$x_M(k+1) = A_M(k)x_M(k) + B_M(k)u_M(k) \qquad (2)$$
$$k = 0,1,2,\dots,p-1$$

which starts from the same initial condition $x_M(0)$ each repetition. The n_s order model is allowed to be time-varying if desired. Since the learning control signals will cause convergence of the system to the desired trajectory, the reference model does not have to represent the desired system behavior in any sense.

Conversion to the Repetition Domain

We convert both the system (1) and the reference model (2) to the repetition domain. Note that

$$x(k) = \phi(k,0)x(0) + \sum_{i=0}^{k-1}\phi(k,i+1)[B(i)u(i)+w(i)]$$
$$\phi(k,i) = \begin{cases} A(k-1)A(k-2)\dots A(i) & k > i \\ I & k = i \end{cases} \qquad (3)$$

and define the full state and control histories for the p-step process by

$$\underline{x}^F = [x^T(1)\ x^T(2)\ \dots\ x^T(p)]^T \qquad \underline{u}^T = [u^T(0)\ u^T(1)\ \dots\ u^T(p-1)]^T$$
$$\underline{x}^F_M = [x^T_M(1)\ x^T_M(2)\ \dots\ x^T_M(p)]^T \qquad \underline{u}^T_M = [u^T_M(0)\ u^T_M(1)\ \dots\ u^T_M(p-1)]^T$$

where a subscript j will be added to indicate the repetition number $j = 0, 1,2,\dots$ Also, define the difference operator $\delta_j(\cdot)$ such that $\delta_j z = z_j - z_{j-1}$ for any appropriate matrix z. Note that $\delta_j x(0) = 0$, $\delta_j w(i) = 0$, so that the set of equations (3) for all k can be written in a matrix form as $\delta_j \underline{x}^F = P_{SF}\delta_j \underline{u}$, and similarily for $\delta_j \underline{x}^F_M$. Then the state equations, in the repetition domain, for the system and the reference model are

$$\underline{x}^F_{j+1} = I\,\underline{x}^F_j + P_{SF}\delta_{j+1}\underline{u}$$
$$\underline{x}^F_{M,j+1} = I\,\underline{x}^F_{M,j} + P_{MF}\delta_{j+1}\underline{u}_M \qquad (4)$$

$$P_{SF} = \begin{bmatrix} B(0) & & & & \\ \phi(2,1)B(0) & B(1) & & \underline{0} & \\ \vdots & \vdots & & & \\ \phi(p,1)B(0) & \phi(p,2)B(1) & \dots & \phi(p,p-1)B(p-2) & B(p-1) \end{bmatrix}$$

with P_{MF} defined analogously.

Feasible Trajectories and the Reduced State Equations

In a digital system, when the dimension of the control vector is less than that of the state, the space spanned by Bu limits the set of states x(k+1) which can be reached from the state x(k). When specifying the desired trajectory, it is important not to specify a state history that violates these limitations. Furthermore, we will limit ourselves to trying to get measured states to track a desired history. Therefore we will eliminate various components of $\underline{x}^F$ (and the coresponding components of $\underline{x}^F_M$), defining a reduced state vector $\underline{x}$ (and $\underline{x}_M$). The corresponding rows are eliminated in P_{SF} and P_{MF} to produce the reduced state equations

$$\begin{aligned} \underline{x}_{j+1} &= I\,\underline{x}_j + P_s\,\delta_{j+1}\,\underline{u} \\ \underline{x}_{M,j+1} &= I\,\underline{x}_{M,j} + P_M\,\delta_{j+1}\,\underline{u}_M \end{aligned} \tag{5}$$

which are of dimension n. The desired system response is then given as a desired value of the reduced state history vector $\underline{x}$ denoted by $\underline{x}^*$. The choice of states that remain in x will be required to be feasible in the following sense.

Definition: The reduced state variable history $\underline{x}$ is said to be feasible for system (1) (or for system (2)) if:

1. Each component in $\underline{x}$ corresponds to a measured quantity.
2. P_s (or P_M) is square and full rank.

Note that if the state variables present in $\underline{x}$ are feasible for (1) (or (2)) then there exists a $\delta_{j+1}\underline{u}$ (or $\delta_{j+1}\underline{u}_M$) that will produce $\underline{x}_{j+1} = \underline{x}^*$ (or $\underline{x}_{M,j+1} = \underline{x}^*$) for all possible choices of $\underline{x}^*$.

Although the definition of feasible $\underline{x}$ for (1) is given in terms of P_s (or equivalently A,B) which is not known, one can nevertheless know that a choice of the reduced state variables is feasible. Consider time invariant systems. The standard results in deadbeat control indicate that:

1. If (1) represents a controllable scalar input, scalar output system, then a reduced state history specifying the full state every n_s steps (with p an integer multiple of n_s) is guaranteed to be feasible.
2. If the number of (nonredundant) inputs, m ,is equal to the dimension of the state n_s ,and the system is controllable, then $\underline{x} = \underline{x}^F$ is a feasible history.

Model Reference Learning

The learning control for the reference model is updated at each repetition of the process by a linear combination of the error history observed in the previous repetition

$$\begin{aligned} \delta_{j+1}\,\underline{u}_M &= \mathcal{L}_M\,\underline{e}_{M,j} \\ \underline{e}_{M,j} &= \underline{x}^* - \underline{x}_{M,j} \end{aligned} \tag{6}$$

From (5),

$$\underline{e}_{M,j+1} = (I - P_M\mathcal{L}_M)\underline{e}_{M,j} \tag{7}$$

and $\lim_{j\to\infty}\underline{x}_{M,j} = \underline{x}^*$,provided the eigenvalues of $(I - P_M\mathcal{L}_M)$ are all within the unit circle. One can pick $\mathcal{L}_M = P_M^{-1}$,in which case learning is accomplished in one repetition, and all eigenvalues of $(I - P_M\mathcal{L}_M)$ are at the origin. Various convenient ways to pick $\mathcal{L}_M$ together with a discussion of the learning rate can be found in [25].

Redefinition of the Model Reference Learning Variable

The Liapunov based learning process developed in the sequel requires an asymptotically stable reference model in the repetition domain. To accomplish this, write

$$\delta_{j+1}\underline{u}_M = G\,\underline{x}_{M,j} + H\,\delta_{j+1}\underline{v}_M \tag{8}$$

and let $\delta_{j+1}\underline{v}_M$ be the new learning variable. When substituted into (5)

$$\begin{aligned} \underline{x}_{M,j+1} &= G_M\,\underline{x}_{M,j} + H_M\delta_{j+1}\underline{v}_M \\ G_M &= I + P_M G \\ H_M &= P_M H \end{aligned} \tag{9}$$

whose response will be identical to that of the original system provided the $\delta_{j+1}\underline{v}_M$used corresponds to $\delta_{j+1}\underline{u}_M = \mathcal{L}_M\,\underline{e}_{M,j}$,or

$$\delta_{j+1}\underline{v}_M = H^{-1}\,[\,\mathcal{L}_M(\underline{x}^* - \underline{x}_{M,j}) - G\,\underline{x}_{M,j}] \tag{10}$$

This requires that H^{-1} exists and that G be such that all $|\lambda_i(I+P_M G)| < 1$.

THE LIAPUNOV BASED MODEL REFERENCE LEARNING CONTROL ALGORITHM

Set up

As discussed above, pick a reference model, choose a feasible reduced state variable history for the model, and prescribe the desired value of this history $\underline{x}^*$. Pick the initial condition $x_M(0)$ from which $\underline{x}_{M,0}$ is obtained, and choose a learning control matrix $\mathcal{L}_M$ for asymptotically stable learning. Pick G to place the eigenvalues of $(I+P_M G)$ within the unit circle, and pick an n×n invertible matrix H. In addition, choose an invertible n×n matrix E_1, an n×n matrix F_1 , and a positive definite matrix R whose maximum eigenvalue is denoted λ_{max}.

Computation of the Learning Control

The learning control is

$$\underline{u}_{j+1} = \underline{u}_j + E_{j+1}F_{j+1}\,\underline{x}_j + E_{j+1}\,\delta_{j+1}\,\underline{v}_M \tag{11}$$

$$j = 1,2,\ldots$$

which can be applied starting at $\underline{u}_2$. The $\underline{x}_j$ is measured. The $\delta_{j+1}\underline{v}_M$ is given in terms of $\underline{x}_{M,j}$ by (10), and $\underline{x}_{M,j}$ is computed from (5). The coefficients are updated according to

$$\begin{aligned} F_{j+1} &= F_j + \Delta\Theta_{F,j} \\ E_{j+1} &= [\,I - E_j\,\Delta\Theta_{E,j}\,]^{-1}\,E_j \end{aligned} \tag{12}$$

$$[\Delta\Theta_{F,j} \,\vdots\, \Delta\Theta_{E,j}] = \left\{ \frac{\alpha_j\,R\,H_M^{-1}[(\underline{x}_{M,j}-\underline{x}_j) - G_M(\underline{x}_{M,j-1}-\underline{x}_{j-1})]}{\lambda_{max}\,[\underline{x}_{j-1}^T\underline{x}_{j-1} + (\underline{u}_j-\underline{u}_{j-1})^T(\underline{u}_j-\underline{u}_{j-1})]} \right\} [\,\underline{x}_{j-1}^T \,\vdots\, (\underline{u}_j-\underline{u}_{j-1})^T]$$

where the parameter α_j is chosen at each repetition step to be in the range of $0 < \alpha_j < 2$, and such that

$$\alpha_j^{-1} \neq \lambda_i\left(\frac{E_j R\,H_M^{-1}\,[(\underline{x}_{M,j}-\underline{x}_j) - G_M(\underline{x}_{M,j-1}-\underline{x}_{j-1})]\,\delta_j\underline{u}^T}{\lambda_{max}\,[\underline{x}_{j-1}^T\underline{x}_{j-1} + (\underline{u}_j-\underline{u}_{j-1})^T(\underline{u}_j-\underline{u}_{j-1})]} \right) \tag{13}$$

for $i = 1,2,\ldots,n$.

CONVERGENCE

The basic result can be stated in the form of a theorem.

Theorem: Given system (1) whose order n_s is known, but whose coefficients A(k), B(k) and whose repetitive disturbance w(k) are unknown. Generate a reference model in accordance with the above algorithm, and prescribe the desired reduced state history $\underline{x}^*$ that is feasible for the model. Provided $\underline{x}^*$ is feasible for (1), application of the learning control (11) will cause x(k) to converge to the desired trajectory $\underline{x}^*$ as the repetitions increase, i.e.

$$\lim_{j \to \infty} \underline{x}_j = \underline{x}^* \tag{14}$$

The proof is developed in the following subsections.

Redefinition of the System Input Variable

In the same manner that (8) was used to stabilize the reference model, we construct $\delta_{j+1}\underline{u}$ according to

$$\delta_{j+1}\underline{u} = E_{j+1} F_{j+1} \underline{x}_j + E_{j+1} \delta_{j+1} \underline{v}_M \tag{15}$$

where E_{j+1}, F_{j+1} are yet to be determined, and the $\delta_{j+1}\underline{v}_M$ input is the same as was used in the reference model (9). Substitution into (5) gives

$$\begin{aligned} \underline{x}_{j+1} &= G_{j+1} \underline{x}_j + H_{j+1} \delta_{j+1} \underline{v}_M \\ G_{j+1} &= I + P_s E_{j+1} F_{j+1} \\ H_{j+1} &= P_s E_{j+1} \end{aligned} \tag{16}$$

In order for this system state $\underline{x}_j$ to converge to the state $\underline{x}_{M,j}$, there must exist values of E_{j+1}, F_{j+1}, call them E^*, F^*, which would cause the matrices G_{j+1}, H_{j+1} in (16) to be identical to the G_M, H_M of the reference model (9). These values exist and are given by

$$\begin{aligned} G_M &= I + P_s E^* F^* \qquad & F^* &= H_M^{-1} (G_M - I) \\ H_M &= P_s E^* & E^* &= P_s^{-1} H_M \end{aligned} \tag{17}$$

where P_s^{-1} exists because $\underline{x}^*$ is feasible, and H_M^{-1} exists, H^{-1}, P_M^{-1} exist.

The Error Equation

By construction, the reference model output converges to the desired path as the repetitions progress, and we wish to make the system output converge to the reference path, and hence the desired path. The error in following the reference path is $\underline{e}_j = \underline{x}_{M,j} - \underline{x}_j$, and we wish to obtain the dynamic equation for this error.

Substract (16) from (9), and substitute $\delta_{j+1}\underline{v}_M = - F_{j+1}\underline{x}_j + E_{j+1}^{-1}\delta_{j+1}\underline{u}$ (from (15)) and G_M from (17) to obtain the error equation.

$$\underline{e}_{j+1} = G_M \underline{e}_j + H_M(F^* - F_{j+1})\underline{x}_j + H_M(E_{j+1}^{-1} - E^{*-1}) \delta_{j+1}\underline{u} \tag{18}$$

A Problem Reminiscent of System Identification

Equation (18) can be rewritten as

$$f_{j+1} = \Theta_{j+1} y_j \tag{19}$$

$$f_{j+1} = H_M^{-1}(\underline{e}_{j+1} - G_M\,\underline{e}_j)$$

$$\theta_{j+1} = [F^* - F_{j+1} \quad E_{j+1}^{-1} - E^{*-1}]$$

$$y_j^T = [\underline{x}_j^T \quad \delta_{j+1}\,\underline{u}^T]$$

As in an identification problem, we know from data the values of f_{j+1} and y_j but we do not know θ_{j+1}. In identification we want to pick θ_{j+1} to eliminate the equation error. In this case, we would like to force f_{j+1} to go to zero. If f_{j+1} approaches zero, then $\underline{e}_{j+1}$ must also, because G_M is an asymptotically stable system matrix by construction. Pick the update equation for θ_{j+1} as

$$\theta_{j+1} = \theta_j - \frac{\alpha_j\, R\, f_j\, y_{j-1}^T}{\lambda_{max}\, y_{j-1}^T\, y_{j-1}} \tag{20}$$

Establishing Convergence by Liapunov Methods

Consider the following as a Liapunov function

$$V_j = \operatorname{tr}(\theta_j^T \theta_j) \tag{21}$$

and study the change in V_j with j

$$\begin{aligned}\Delta V_j &= V_{j+1} - V_j \\ &= \operatorname{tr}(\theta_{j+1}^T \theta_{j+1} - \theta_j^T \theta_j) \\ &= \operatorname{tr}[(-2\varepsilon_j \theta_j^T R + \varepsilon_j^2 y_{j-1} f_j^T R^2) f_j y_{j-1}^T] \\ &= y_{j-1}^T[-2\varepsilon_j \theta_j^T R + \varepsilon_j^2 y_{j-1} f_j^T R^2] f_j \\ &= -f_j^T Q\, f_j \end{aligned} \tag{22}$$

where $\varepsilon_j = \alpha_j/(\lambda_{max} y_{j-1}^T y_{j-1})$, and where the following identity has been used: $\operatorname{tr}(Myx^T) = x^T My$ for M symmetric and x, y column vectors. We want the matrix Q to be positive definite

$$\begin{aligned} Q &= \varepsilon_j [2R - \varepsilon_j (y_{j-1}^T\, y_{j-1})R^2] \\ &= \varepsilon_j R^{1/2}[2I - \varepsilon_j (y_{j-1}^T\, y_{j-1})R]\, R^{1/2} \\ &= \varepsilon_j R^{1/2}[2I - \frac{\alpha_j}{\lambda_{max}} R]\, R^{1/2} \end{aligned} \tag{23}$$

where $R^{1/2}$ is a matrix such that $R^{1/2} R^{1/2} = R$. It can be obtained from the eigendecomposition of $R = P^T \operatorname{diag}(\lambda_1, \dots, \lambda_n)\, P$ (which always exists since R is symmetric) by writing $R^{1/2} = P^T \operatorname{diag}(\lambda_1^{1/2}, \dots, \lambda_n^{1/2})\, P$. The eigenvectors of the square bracket in (23) are those of R, and post multiplying by this vector shows the eigenvalues to be $(2 - \alpha_j \lambda_i/\lambda_{max})$ which are always positive provided $\alpha_j < 2$. Also $\alpha_j > 0$ is needed to make the ε_j in (23) positive. Hence, when α_j is chosen to satisfy $0 < \alpha_j < 2$, Q is positive definite, and therefore ΔV_j is negative for all j provided $f_j \neq 0$.

The Liapunov function at step j can be written as

$$V_0 + \Delta V_0 + \Delta V_1 + \dots + \Delta V_{j-1} > 0 \tag{24}$$

and using $V_j > 0$, $V_0 > 0$ shows that $V_0 > -\Delta V_0 - \Delta V_1 - \dots - \Delta V_{j-1} > 0$. If f_j becomes identically zero in a finite number of steps, we have accomplished our goal of making $\underline{e}_j$ become zero. If f_j is never zero, then the sum $(-\Delta V_0)+\ldots+(-\Delta V_{j-1})$ becomes an infinite sum of positive terms and hence it must converge. By (22), if $\Delta V_j \to 0$, then $f_j \to 0$, and again we have the desired result.

The Control Gains

To determine how to update the gains E , F , use (19) and (20) to write

$$\theta_{j+1} - \theta_j \triangleq - [\Delta\theta_{F,j}, \Delta\theta_{E,j}] = - [F_{j+1} - F_j, -E_{j+1}^{-1} + E_j^{-1}]$$
$$= - \alpha_j R f_j y_{j-1}^T / (\lambda_{max} y_{j-1}^T y_{j-1}) \qquad (25)$$

from which

$$F_{j+1} = F_j + \Delta\theta_{F,j}$$
$$E_{j+1}^{-1} = E_j^{-1} - \Delta\theta_{E,j} \qquad (26)$$
$$E_{j+1} = [E_j^{-1} - \Delta\theta_{E,j}]^{-1} = [I - E_j \Delta\theta_{E,j}]^{-1} E_j$$

The last line solving for E_{j+1} requires that $[I - E_j \Delta\theta_{E,j}]^{-1}$ exists. Its determinant is

$$\det [I - E_j \Delta\theta_{E,j}] = \alpha_j^n \det \left[\frac{1}{\alpha_j} I - \frac{E_j R f_j \delta_j \underline{\mu}^T}{\lambda_{max} y_{j-1}^T y_{j-1}} \right] \qquad (27)$$

which is nonzero provided $1/\alpha_j$ is not an eigenvalue of the second term in the square bracket. This term is known at the time α_j must be picked, so that one can always insure existence of the inverse.

The equations (25-27) establish the conditions (12,13) of the algorithm, and the convergence of the learning process is established.

CONCLUSIONS

We have generated a learning control algorithm which is direct in the sense that it does not involve system identification, and the algorithm is guaranteed to converge whenever the desired trajectory can be specified in a manner known to be feasible for the system. This happens, for example, in any controllable systems with a scalar control such as single-input single-output systems, or in any controllable systems with as many independent controls as states. Further work will generalize the results to broader classes of systems.

REFERENCES

1. M.A. Aizerman et al., "Potential Functions Technique and Extrapolation in Learning Systems Theory", Proc. 3rd IFAC Congress, Butterworth, London, 1966.
2. Ya. Z. Tsypkin, Adaptation and Learning in Automatic Systems, Academic Press, New York, 1971.
3. K. Fu, "Learning Control Systems--Review and Outlook", IEEE Transactions on Automatic Control, Vol. AC-15, pp. 210-221, April 1970.
4. M. Uchiyama, "Formation of High Speed Motion Pattern of Mechanical Arm by Trial", Trans. Soc. Instr. Contr. Eng., Vol.19, No. 5, 1978, pp. 706-712.

5. S. Arimoto, "Mathematical Theory of Learning with Applications to Robot Control," Proceedings of the Fourth Yale Workshop on Applications of Adaptive Systems Theory, May 1985.
6. S. Arimoto, S. Kawamura, F. Miyazaki, and S. Tamaki, "Learning Control Theory for Dynamical Systems", Proceedings of the 24th IEEE Conference on Decision and Control, Fort Lauderdale, Fla. December 1985.
7. S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering Operation of Robots by Learning," Journal of Robotic Systems, Vol. 1, No. 2, pp. 123-140, 1984.
8. S. Arimoto, S. Kawamura, and F. Miyazaki, "Can Mechanical Robots Learn by Themselves?", Robotics Research, The Second International Symposium (ed. by H. Hanafusa and H. Inoue), MIT Press, Cambridge, Mass., 1985, pp. 127-134.
9. S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering Operation of Dynamical Systems by Learning: A New Control Theory for Servomechanisms or Mechatronics Systems", Proceedings of the 23rd IEEE Conference on Decision and Control, Las Vegas, Nevada, Dec. 1984.
10. S. Kawamura, F. Miyazaki, and S. Arimoto, "Iterative Learning Control for Robotic Systems", Proceedings of IECON '84, Tokyo, Japan, Oct.84.
11. S. Kawamura, F. Miyazaki, and S. Arimoto, "Applications of Learning Methods for Dynamic Control of Robot Manipulators", Proceedings of the 24th Conference on Decision and Control, Fort Lauderdale, Fla., Dec. 1985.
12. S. Hara, T. Omata, and M. Nakano, "Synthesis of Repetitive Control Systems and Its Applications", Proceedings of the 24th Conference on Decision and Control, Fort Lauderdale, Fla. Dec 85.
13. M. Togai and O. Yamano, "Analysis and Design of an Optimal Learning Control Scheme for Industrial Robots: A Discrete-Time Approach", Proceedings of the 24th Conference on Decision and Control, Fort Lauderdale, Fla., Dec. 1985.
14. T. Mita and E. Kato, "Iterative Control and Its Applications to Motion Control of Robot Arm--A Direct Approach to Servo Problems", Proceedings of the 24th Conference on Decision and Control, Fort Lauderdale, Fla., Dec. 1985.
15. K. Furuta and M. Yamakita, "Iterative Generation of Optimal Input of a Manipulator", Proceedings of the IEEE Conference on Robotics, 1986.
16. E.G. Harokopos, "Optimal Learning Control of Mechanical Manipulators in Repetitive Motions", Proceedings of the IEEE Conference on Robotics '86
17. G. Casalino and L. Gambardella, "Learning of Movements in Robotic Manipulators", Proceedings of the IEEE Conference on Robotics, 1986.
18. C.G. Atkeson and J. McIntyre, "Robot Trajectory Learning Through Practice", Proceedings of the IEEE Conference on Robotics and Automation, San Francisco, 1986.
19. J.J. Craig, "Adaptive Control of Manipulators Through Repeated Trials", General Motors Research Report GMR-4530, Dec. 1983, and Proceedings of the American Control Conference, San Diego, June 1984.
20. S.H. Wang, "Computed Reference Error Adjustment Technique (CREATE) for the Control of Robot Manipulators", 22nd Annual Allerton Conference on Communication, Control, and Computing, October 1984.
21. S.H. Wang and I. Horowitz, "CREATE--A New Adaptive Technique", Proceedings of the 19th Annual Conference on Information Sciences and Systems, March 1985.
22. E. Harokopos and R.W. Longman, "A Learning Controller for Tracking Arbitrary Inputs in Robot Motions", Tenth IASTED International Symposium on Robotics and Automation, Lugano, Switzerland, June 29-July 2, 1987.

23. R.H. Middleton, G.C. Goodwin and R.W. Longman, "A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task", accepted for publication in The International Journal of Robotics Research.
24. F. Pourboghrat, "Adaptive Learning Control for Robots", Proceedings of the IEEE Conference on Robotics and Automation, 1988.
25. M. Phan and R.W. Longman, "A Mathematical Theory of Learning Control for Linear Discrete Multivariable Systems", Proceedings of the AIAA/AAS Astrodynamics Specialist Conference, Minneapolis, Minnesota, August 1988.
26. K.S. Narendra and P. Kudva, "Stable Adaptive Schemes for System Identification and Control - Part I", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC No. 6, November 1974.
27. P. Kudva and K.S. Narendra, "An Identification Procedure for Discrete Multivariable Systems", IEEE Transactions on Automatic Control, Vol. AC-18, October 1973.
28. M. Phan and R.W. Longman, "Indirect Learning Control with Guaranteed Stability", to appear.

On Linear Systems Driven by Self-Excited Point Processes*

Mary Ann Ingram
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250

Abraham H. Haddad
Department of EE/CS
Northwestern University
Evanston, Illinois 60208

ABSTRACT

In this work, mean-square continuity is proved for the state of a linear system disturbed by a point process with a state-dependent rate and random marks. The cross-correlation property and the linear optimal filter are derived for the case of zero mean marks.

SUMMARY

The model treated in this summary is a continuous linear time invariant system driven by a self-excited, marked point process. The term "self-excited" implies that the instantaneous average jump rate or intensity of the point process depends on the history of the process. Thus, self-excitation is one kind of time-correlation. In particular, the jump rate is specified as a memoryless function of the system state. The term "marked" describes a point process with random jump amplitudes (marks).

One possible application of this model is in the tracking of maneuvering targets. The jump process represents the commanded acceleration of the vehicle being tracked. The state-dependency of the rate represents a relation between the rate of acceleration jumps and the position and velocity of the vehicle. Another possible application is in state estimation for systems subject to abrupt failures, such as the onset of biases in sensors or actuators. Here, state-dependency of the rate may represent an increased vulnerability to failures under conditions of high heat, speed, or electrical current.

The state process is given by the following stochastic differential equation

$$dx_t = Ax_t dt + BdM_t \quad t \geq 0$$

where $x_t \in \Re^n$ is the state, with initial condition x_0, $A \in \Re^n \times \Re^n$, $B \in \Re^n$, and $M_t \in \Re$ is a piecewise constant random process to be defined below.

Let N_t denote the number of jumps in M_t. Let $\{u_1, u_2, \ldots, u_{N_t}\}$ be the consecutive jump heights or marks of M_t. Then M_t may be expressed

$$M_t = \sum_{i=1}^{N_t} u_i.$$

The marks are assumed to be independently and identically distributed with mean $\overline{u}$ and mean square value $\overline{u^2}$. We define the stochastic intensity [1] or instantaneous average rate of N_t to be $\mu[x_t]$, where μ is a scalar valued, positive, and bounded function of the state x_t of the system. It follows that $N_t - \int_0^t \mu[x_s]ds$ is a martingale with respect to the σ-algebra S_t generated by x_t. The observation model is given by

$$dz_t = x_t dt + dw_t$$

where w_t is an n-vector of Wiener processes with $E\{dw_t dw_t\} = Idt$.

There has been a fair amount of work concerning systems driven by compound Poisson processes, that is, independent increment point processes with random marks. The contribu-

---

*Supported by the U. S. Air Force under Grant AFOSR–87–0308

tions include representations and properties [2], and mean-square optimal, linear optimal, and suboptimal state estimation [3,4]. Self-excited and more general point processes have received attention [2,1], mainly as models for point process observations of dynamic systems.

Martingale theory has been applied successfully in the characterization of point processes [1] as well as in nonlinear filtering theory. Thus, it was desirable and instructive to use it in proving the following propositions.

Proposition 1 *If there exists a constant K such that*

$$\mu[x] < K < +\infty$$

for all $x \in \Re^n$, then x_t is mean-square continuous, that is,

$$\lim_{s \to t} E\{\|x_t - x_s\|^2\} = 0.$$

where $\|\cdot\|$ denotes the Euclidean distance in $\Re^n$.

Proposition 2 *If the mean of the marks is zero, i.e., if $\overline{u} = 0$, then x_t is of the separable class [5], which implies that for any nonlinear, scalar valued function $g(\cdot)$, there exists a constant vector C such that*

$$E\{x_t g(x_\tau)\} = E\{x_t x_\tau'\} C.$$

given that the appropriate expectations exist.

The innovations approach of Kailath [6] may be used to derive the linear filter. The mean-square continuity of x_t is sufficient to prove that the innovations process $v_t = z_t - \int_{t_0}^{t} \hat{x}_s \, ds$, where $\hat{x}_t$ is the filter output, is a process of orthogonal increments. When the marks have zero mean, the resulting filter is

$$\begin{aligned} d\hat{x}_t &= A\hat{x}_t dt + P_t dv_t \\ \dot{P}_t &= AP_t + P_t A' + BB' \, \overline{u^2} \, \overline{\mu[x_t]} - P_t P_t \end{aligned}$$

Although P_t will be difficult to compute due to the a priori expectation $\overline{\mu[x_t]}$, the computation can be done off-line. We are currently investigating methods of computation and the filter expression for the case of nonzero mean marks.

References

[1] P. Brémaud, *Point Processes and Queues, Martingale Dynamics*, Springer-Verlag, New York, 1981.

[2] D. L. Snyder, *Random Point Processes*, Wiley, New York, 1975.

[3] S. P. Au and A. H. Haddad, "Suboptimal sequential estimation-detection scheme for Poisson driven linear systems," *Inform. Sci.*, Vol. **16**, pp. 95-113, 1978.

[4] H. Kwakernaak, "Filtering for systems excited by Poisson white noise," Eds.: A. Bensoussan, J. L. Lions, *Control Theory, Numerical Methods, and Computer Systems*, Springer Lecture Notes in Economics and Mathematical Systems, Vol. **107**, Berlin, pp. 468–492, 1975.

[5] A. H. Haddad, "Dynamical representation of markov processes of the separable class," *IEEE Trans. Information Theory*, Vol. **IT-16**, No. 5, September 1970.

[6] T. Kailath, "A note on least squares estimation by the innovations method," *SIAM J. Contr.*, Vol. **10**, No. 3, pp. 477-486, August 1972.

ADAPTIVE STABILIZATION OF MULTI-MACHINE POWER SYSTEMS

H. YOUSEF and M. SIMAAN
Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261

ABSTRACT

This paper presents a new decentralized adaptive control scheme to enhance the damping characteristics of the electromechanical modes of a multi-machine power system. The control scheme uses the Model Reference Adaptive Control (MRAC) technique. The desired damping performance of the electromechanical modes are specified in the reference model. Asymptotic tracking between the output of each machine and that of each submodel is guaranteed provided that the subsystem transfer matrix is Strict Positive Real (SPR). A typical multi-machine power system is used as a test example to illustrate the effectiveness of the proposed scheme.

I. INTRODUCTION

The problem of improving the damping of machine oscillations by means of supplementary excitation control, termed as power system stabilizer (PSS), has received considerable attention in the literature on power systems. The basis for stabilizing a single machine connected to an infinite bus using a speed deviation signal and suitable ranges of stabilizer parameters are presented in [1]. The enhancement of the damping performance of the electromechanical oscillations in multi-machine power systems has recently become an important issue due to the increasing size and complexity of power systems [2, 3]. The application of power system stabilizers at different locations has been used to damp out these oscillations. Eigenvalue-eigenvector techniques have been used in the analysis, damping of these oscillations [2, 4], and tuning the power system stabilizers parameters [5]. Techniques for determining the parameters of PSS in a multi-machine power system are reported in [6, 7]. A decentralized suboptimal stabilizer design technique to determine the stabilizer parameters is proposed in [8]. Recently, PSS design methods based on adaptive control and self-tuning regulator techniques have been discussed in the literature [9, 10]. However, these works have considered only single-machine infinite-bus models.

The purpose of this paper is to apply the decentralized adaptive control algorithm proposed in [11] to a multi-machine power system in order to improve the damping of the electromechanical oscillations. The control law is designed in such a way as to achieve asymptotic tracking between the outputs of a multi-machine power system and those of a given reference model.

II. MULTI-MACHINE POWER SYSTEM MODEL

A multi-machine power system consisting of N generators and linearized around an operating point corresponding to a condition of equilibrium can be modelled in the form :

$$\dot{x}_i = A_i\, x_i + B_i\, u_i + \sum_{\substack{j=1 \\ j \neq i}}^{N} A_{ij}\, x_j \tag{1}$$

$$y_i = C_i\, x_i \tag{2}$$

for $i=1,2,\ldots,N$, where $x_i = [\Delta\, \omega_i\ ,\ \Delta\, \delta_i\ ,\ \Delta\, E'_{qi},\ \Delta\, V_{fi}\ ]^t$ is the state vector (the superscript t denotes transpose), u_i is the stabilizing control signal and $y_i = \Delta\, \omega_i$ is the speed deviation output signal. The matrices A_i, A_{ij}, B_i and C_i are of compatible dimensions and given by :

$$A_i = \begin{bmatrix} \frac{-D_i}{M_i} & \frac{-K_{1ii}}{M_i} & \frac{-K_{2ii}}{M_i} & 0.0 \\ \omega_o & 0.0 & 0.0 & 0.0 \\ 0.0 & \frac{-K_{4ii}}{\tau_{doi}} & \frac{-1}{K_{3ii}\tau_{doi}} & \frac{1}{\tau_{doi}} \\ 0.0 & \frac{-K_{Ai}K_{5ii}}{T_{Ai}} & \frac{-K_{Ai}K_{6ii}}{T_{Ai}} & \frac{-1}{T_{Ai}} \end{bmatrix},\quad B_i = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ \frac{K_{Ai}}{T_{Ai}} \end{bmatrix},$$

$$A_{ij} = \begin{bmatrix} 0.0 & \frac{-K_{1ij}}{M_i} & \frac{-K_{2ij}}{M_i} & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & \frac{-K_{4ij}}{\tau_{doi}} & \frac{-1}{K_{3ij}\tau_{doi}} & 0.0 \\ 0.0 & \frac{-K_{Ai}K_{5ij}}{T_{Ai}} & \frac{-K_{Ai}K_{6ij}}{T_{Ai}} & 0.0 \end{bmatrix} \quad and\ C_i^t = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

where ω_i is the angular speed, ω_o is the rated angular speed, δ_i is the torque angle, E'_{qi} is the transient voltage proportional to the field flux linkages, V_{fi} is the field voltage, M_i is the inertia coefficient D_i is the machine damping coefficient, τ_{doi} is the d-axis open-circuit time constant, K_{Ai} is the voltage regulator gain, T_{Ai} is the voltage regulator time constant and the index *i* refers to the ith machine. The block diagram of the ith machine is shown in figure 1. The parameters K_{1ii}, K_{1ij},...K_{6ij}, with the exception of K_{3ii} and K_{3ij}, are function of machine loading and the admittance matrix of the transmission system. A procedure to evaluate these parameters is given in [12]. The composite system can be written as:

$$\dot{x} = A\, x + B\, u + H\, x \tag{3}$$

$$y = C\, x \tag{4}$$

where $x^t = [\ x_1^t, x_2^t, \ldots x_N^t\ ]$, $u^t = [\ u_1^t, u_2^t, \ldots u_N^t\ ]$ and $y^t = [\ y_1^t, y_2^t, \ldots y_N^t\ ]$ are the composite state vector, control vector and output vector respectively. The matrices A, B and C are block diagonal. The matrix H has zeroes on the block diagonal, A_{ij} on the upper block diagonal and A_{ji} on the lower block diagonal.

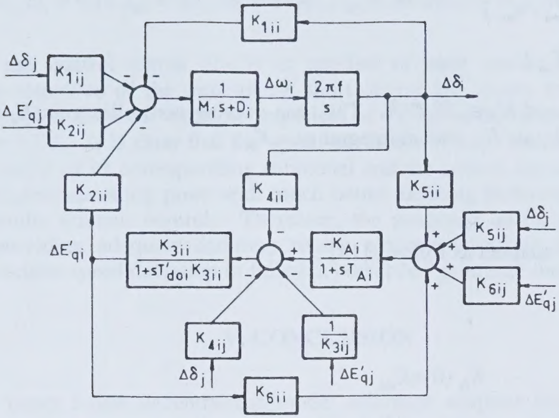

K$_{1ii}$
Δδ$_j$
K$_{1ij}$
ΔE'$_{qj}$
K$_{2ij}$
1
M$_i$s+D$_i$
Δω$_i$
2πf
s
Δδ$_i$
K$_{2ii}$
K$_{4ii}$
K$_{5ii}$
Δδ$_j$
K$_{5ij}$
ΔE'$_{qi}$
K$_{3ii}$
1+sT'$_{doi}$K$_{3ii}$
-K$_{Ai}$
1+sT$_{Ai}$
K$_{6ij}$
ΔE'$_{qj}$
K$_{4ij}$
1
K$_{3ij}$
Δδ$_j$
ΔE'$_{qj}$
K$_{6ii}$

Figure 1. Block Diagram of the ith Machine in a Multi-Machine System

If the speed deviation $y_i = \Delta\ \omega_i$ of each machine is poorly damped and the system parameter and/or the operating point changes, we design an adaptive control scheme so taht y_i tracks asymptotically the output y_{mi} of an SISO submodel having a desired damping characteristic. The ith submodel is written as:

$$\dot{x}_{mi} = A_{mi}\, x_{mi} + B_{mi}\, u_{mi} \tag{5}$$

$$y_{mi} = C_{mi}\, x_{mi} \tag{6}$$

where $x_{mi} \in \Re^{m_i}$. The matrices A_{mi}, B_{mi} and C_{mi} are given constant matrices of appropriate dimensions and the matrix A_{mi} is asymptotically stable for i=1,...,N. Due to the presence of interconnections, it is not possible to ensure that $\lim\limits_{t \to \infty} \|y_i - y_{mi}\| = 0$ for any bounded input vector u_{mi}. The best we can achieve in this case is the convergence of the error vector $(y_i - y_{mi})$ to some finite residual set [13]. The size of the residual set depends on the strength of the interconnection.

III. STRUCTURE OF THE ADAPTIVE CONTROL LAW

The control law u_i of the ith subsystem is defined as a linear combination of the output error vector and the reference model state and input vectors [14] as follows:

$$u_i = K_i\, r_i \tag{7}$$

where

$$r_i^t = [\,(y_{mi}-y_i)^t\,,\, x_{mi}^t\,,\, u_{mi}^t\,] \tag{8}$$

$$K_i = [\,K_{ei}\,,\, K_{xi}\,,\, K_{ui}\,] \tag{9}$$

where $r_i \in \Re^{(m_i+2)}$ and $K_i^t \in \Re^{(m_i+2)}$. The time varying controller gain K_i is chosen as a sum of a proportional gain K_{pi} and an integral gain K_{Ii} :

$$K_i = K_{pi} + K_{Ii} \tag{10}$$

where K_{pi} and K_{Ii} are adapted as follows [14] :

$$K_{pi} = g_i\, r_i^t\, T_i \tag{11}$$

$$\dot{K}_{Ii} = g_i\, r_i^t\, G_i \quad , \qquad K_{Ii}\,(0)=K_{Iio} \tag{12}$$

$$g_i = C_i\, e_i \quad . \tag{13}$$

The error vector e_i in (10)-(12) is defined as :

$$e_i = x_i^* - x_i \tag{14}$$

where x_i^* is the ideal state vector which occurs when the output tracking error (y_i - y_{mi}) = 0 and can be written as a linear combination of the state and input vectors of the submodel [15]. The weighting matrices T_i and G_i in (10) and (11) are selected to be positive definite to guarantee stability. Stability of the error dynamical equations of this adaptive control structure has been studied [11, 13] using the weighted sum scalar Lyapunov function approach. Results of the stability analysis show that the output error as well as the controller parameters are ultimately bounded provided that the decoupled subsystem transfer matrix is SPR.

VI. NUMERICAL EXAMPLE

The nine-bus power system consisting of three machines [16] shown in figure 2 is used as an illustrative example. The steady state power flows are shown in figure 2 and the generators data is given in table 1. It is assumed that the system is subjected to a three-phase fault occurring near bus 7 at the end of line 5-7 and the fault is cleared in five cycles by opening the circuit breakers at both ends of line 5-7. The reduced 3x3 admittance matrices for the prefault, faulted and postfault conditions are given in table 2. The speed deviation $\Delta\, \omega_i$ of each machine (without control) is shown in figure 3. The need for a supplementary control signal to damp out the speed oscillation is obvious. The adaptive control scheme given in equations (7-13) is applied to each machine in order to improve the damping of the speed deviations. A desired damping performance of the electromechanical oscillation mode is specified in the following transfer function of a 2nd order reference model :

$$G_{mi} = \frac{y_{mi}}{u_{mi}} = \frac{k_i\, s}{s^2 + 2\,\zeta_i\,\omega_i\, s + \omega_i^2} \tag{15}$$

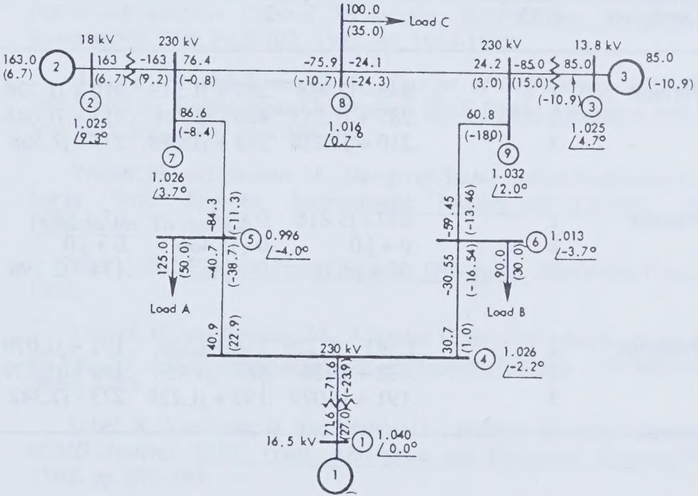
where k_i, ζ_i and ω_i are chosen according to a desired specification. In this example, the following values of ζ_i and ω_i are selected :

$\zeta_1 = .34$, $\omega_1 = 6.15$,$\zeta_2 = .45$, $\omega_2 = 11.47$, $\zeta_3 = .46$ and $\omega_3 = 13.42$.

When the control signal (7-13) is applied to each machine, simulation results following the clearance of the three-phase short circuit are shown in figures 4, 5 and 6. Figures 4 through 6 show the dynamic responses of the speed deviation $\Delta\ \omega_i$ and the model outputs y_i, $i = 1,2,3$. It is clear that the speed deviation of each machine is asymptotically tracking the output of its corresponding submodel and the control signal brings the machine back to its original operating point with much better damping performance as compared to simulation results without control. Therefore, the proposed adaptive control scheme is capable of providing adequate damping to the electromechanical oscillation modes by forcing the machine speed deviation to follow a prescribed reference model.

V. CONCLUSION

In this paper a new decentralized model reference adaptive control algorithm for a class of interconnected systems discussed in [11, 13] has been applied to a multi-machine power system to enhance the damping performance of the electromechanical modes. In order to ensure stability of the output error, the control scheme requires that the decoupled subsystem transfer matrix be strict positive real. A typical power system is used as an illustrative example to demonstrate the effectiveness of the control scheme. Simulation results of the system reveals the capability of the proposed control scheme to provide adequate damping for the system.



18 kV
230 kV
163.0
(6.7)
2
163
-163
76.4
(6.7)
(9.2)
(-0.8)
2
1.025
∠9.3°
86.6
(-8.4)
7
1.026
∠3.7°
100.0
(35.0)
Load C
-75.9
-24.1
(-10.7)
(-24.3)
8
1.016
∠0.7°
230 kV
13.8 kV
24.2
-85.0
85.0
(3.0)
(15.0)
(-10.9)
3
85.0
(-10.9)
3
1.025
∠4.7°
60.8
(-18.0)
9
1.032
∠2.0°
-84.3
(-11.3)
5
0.996
∠-4.0°
125.0
(50.0)
-40.7
(-38.7)
Load A
-59.45
(-13.46)
6
1.013
∠-3.7°
-30.55
(-16.54)
90.0
(30.0)
Load B
40.9
(22.9)
230 kV
30.7
(1.0)
4
1.026
∠-2.2°
-71.6
(-23.9)
71.6
(27.0)
16.5 kV
1
1.040
∠0.0°
1
71.6
(27)

Figure 2. Nine-bus, Three-machine Power System (From [16])

Table 1. Three-Machine System Data (From [16])

Quantity	Gen 1 Hydro	Gen 2 Steam	Gen 3 Steam
H	23.64	6.4	3.01
Rated MVA	247.5	192.	128.
Rated KV	16.5	18.	13.8
x_d	.146	.8958	1.3125
x_d'	.0608	.1198	.1813
x_q	.0969	.8645	1.2578
x_q'	.0969	.1969	.25
τ_{do}	8.96	6.	5.89
K_A	100.	100.	200.
T_A	.05	.05	.05

Table 2. Reduced Y matrix (From [16])

Condition	Node	1	2	3
Prefault	1	.846 - j2.988	.287 + j1.513	.210 + j1.226
	2	.287 + j1.513	.420 - j2.724	.213 + j1.088
	3	.210 + j1.226	.213 + j1.088	.277 - j2.368
Faulted	1	.657 - j3.816	.0 + j.0	.07 + j.631
	2	.0 + j.0	.0 - j5.486	.0 + j.0
	3	.07 + j.631	.0 + j.0	.174 - j2.796
Postfault	1	1.181 - j2.229	.138 + j.726	.191 + j1.079
	2	.138 + j.726	.389 - j1.953	.199 + j1.229
	3	.191 + j1.079	.199 + j1.229	.273 - j2.342

REFERENCES

1. deMello, F. P. and Concordia, C., *Concepts of Synchronous Machine Stability as Affected by Excitation Control*, IEEE Trans. Power Apparatus and Systems,Vol. Vol. PAS-88, 1969, pp. 316-329.

2. deMello, F. P., Nolan, P. J., Laskowsk, T. F. and Undrill, J. M. , *Coordinated Application fo Stabilizers in Multimachine Power Systems*, IEEE Trans. Power Apparatus and Systems,Vol. Vol. PAS-99, 1980, pp. 892-901.

3. Castro, J. C., Catao, M. O. and Doraiswami, R., *Identification of the Generating Units to be Equipped with Stabilizers in a Multimachine Power Systems*, Automatica,Vol. 24, 1988, pp. 405-409.

4. Arcidiacono, V., Ferrari, E., Marconnoto, R., Dos Ghali, J. and Grandez, D., *Evaluation and Improvement of Electromechanical Oscillation Damping by Means of Eigenvalue-Eigenvector Analysis* , IEEE Trans. Power Apparatus and Systems,Vol. PAS-99, 1980, pp. 769-778.

5. Lefebvre, S., *Tuning of Stabilizers in Multimachine Power Systems*, IEEE Trans. Power Apparatus and Systems,Vol. PAS-102, 1983, pp. 290-299.

6. Fleming, R. J., Mohan, M. A. and Parvatisan, *Selection of Parameters of Stabilizers in Multimachine Power Systems*, IEEE Trans. Power Apparatus and Systems,Vol. Vol. PAS-100, 1981, pp. 2329-2333.

7. Gooi, H. B., Hill, E. F., Mobarak, M. A., Thorne, D. H. and Lee T. H. , *Coordinated Multi-Machine stabilizer Settings Without Eigenvalue Drift*, IEEE Trans. Power Apparatus and Systems,Vol. PAS-100, 1981, pp. 3879-3887.

8. Xu, W. and Lu, Q., *A New Stabilizer Design Technique For Multimachine Power Systems*, Electric Power Systems Research,Vol. 15, 1988, pp. 89-97.

9. Ghosh, A., Ledwich, G., Malik, O. P. and Hope, G.S., *Power System Stabilizer Based on Adaptive Control Techniques*, IEEE Trans. Power Apparatus and Systems,Vol. Vol. PAS-103, 1984, pp. 1983-1989.

10. Hsu, Y. Y. and Liou, K. L., *Design of Self-Tuning PID Power System Stabilizers for Synchronous Generators*, IEEE Power Apparatus and Systems,Vol. Vol. PAS-106, 1987 , pp. 343-348.

11. Yousef, H. and Simaan, M., *Design of Stable Model Following Controllers for Large Scale Systems*, International Journal of Control, Accepted for Publication, To Appear.

12. Yu, Y. N., Electric Power System Dynamics, Academic Press, New York, 1983.

13. Yousef, H. and Simaan, M., *Adaptive Control of a Large-Scale System*, Proc. of the 3rd IEEE Symposium on Intelligent Control , Arlington, VA, Aug. 24-26, 1988 .

14. Sobel, K.,Kaufman, H. and Mabius, L., *Implicit Adaptive Control for a Class MIMO Systems*, IEEE Trans. Aerospace and Electronic Systems,Vol. AES-18, 1982, pp. 576-589.

15. O'Brien, M. J. and Broussard, J. R., *Feedforward Control to Track the Output of a Forced Model*, IEEE Trans. Automatic Control,Vol. AC-25, 1980, pp. 851-853.

16. Anderson, P. M. and Fouad, A. A., Power System Control and Stability, Iowa State University Press, Ames, Iowa, 1977.

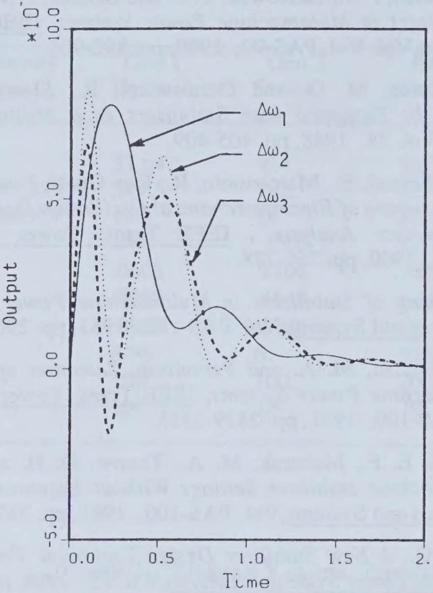


Figure 3. Dynamic Response of The Speed Deviation of Each Machine Following a Three-Phase Short Circuit Without Control

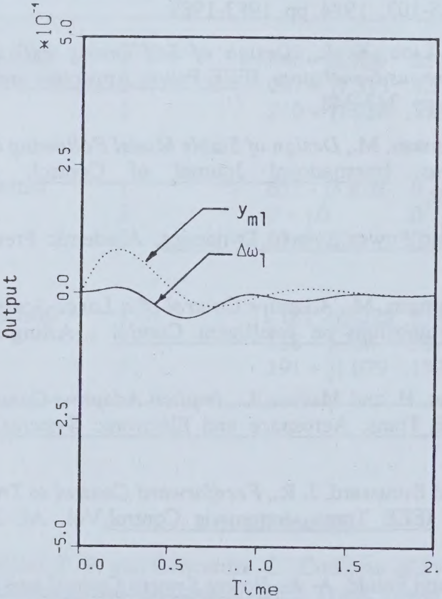


Figure 4. Speed deviation of Machine 1 and Output of Reference model 1 Following A Three-Phase Short Circuit with Control

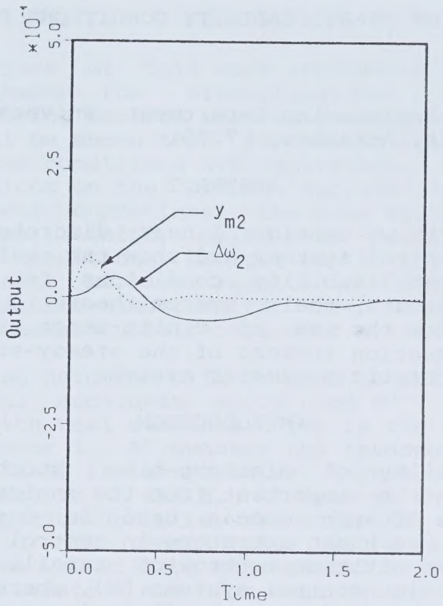


Figure 5. Speed deviation of Machine 2 and Output of Reference model 2 Following A Three-Phase Short Circuit with Control

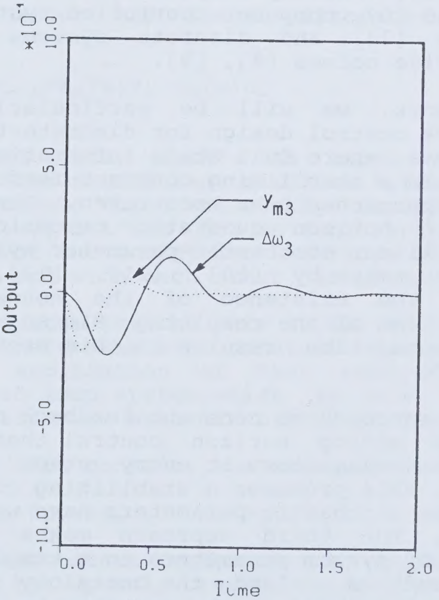


Figure 6. Speed deviation of Machine 3 and Output of Reference model 3 Following A Three-Phase Short Circuit with Control

THE EQUIVALENCE OF STABILIZABILITY CONDITIONS FOR STOCHASTIC SYSTEMS

ENGIN YAZ
Electrical Engineering Department, University of Arkansas
Fayetteville, Arkansas, 72701

ABSTRACT

In this work, we consider linear discrete-time stochastic-parameter control systems and show the equivalence between mean square stabilizability conditions for two different controllers. Besides their system theoretic importance, the results will allow the use of finite-stage solutions of a Riccati-like equation instead of the steady-state solution in stabilizing stochastic-parameter systems.

INTRODUCTION

Stabilizability of discrete-time stochastic-parameter control systems is important from the engineering viewpoint due to abundance of such models used in describing random phenomena such as human operators in control task [1], or in applications like attitude control of satellites and spacecraft and missile control systems [2], where it is becoming more and more widespread to use digital controllers. Some other application possibilities include macroeconomic systems as proposed by [3], randomly-sampled systems which are treated in [4], [5], and [6], computer-controlled systems with random rounding errors [7], and discrete systems with randomly correlated additive noises [8], [9].

In this work, we will be particularly interested in constant-feedback control design for discrete-time stochastic-parameter systems where full state information is available. In order to obtain a stabilizing constant-feedback controller, mainly three approaches have been taken. The first approach uses the infinite horizon quadratic criterion minimization subject to the linear stochastic-parameter system constraint. This approach was taken by [10], [11], [12], who proposed conditions for the existence of the nonnegative definite stabilizing solution of the resulting Riccati-like equation. For the continuous-time results in this area, the reader is referred to [13].

The second approach to constant feedback control of such systems is the moving horizon control that is obtained by minimizing a fixed-stage cost at every stage of the system operation [14]. This produces a stabilizing constant-feedback controller if the stochastic-parameters have weakly stationary characteristics. The third approach makes use of only the expected values of system parameters in a completely deterministic design and is called the certainty (deterministic)-equivalent control method which requires a more stringent bound on the allowable noise powers to secure stochastic stability [15], [16].

The objectives of this work are threefold. First, the relationship between the stabilizability conditions of infinite and moving horizon control methods will be investigated and it will be shown that for mean square observable systems, these two conditions are equivalent. Second, we will relax the conditions on the moving horizon controllers and prove under these conditions, the mean square stability of these controllers. Third, using the equivalence of the stabilizability conditions brought out in this work, we will derive the infinite horizon results of [12], as natural extentions of finite-time results. Then, we will specialize these results to deterministic-parameter systems.

The following notation will be used in this paper. R^n is the n-dimensional euclidean space and R^{nxm} is the space of (nxm) matrices with real elements. $\|x\|$ is the euclidean norm $((x^T x)^{\frac{1}{2}})$ of vector x. A^T denotes the transpose of matrix A. $A>B$ $(A\geq B)$ means $(A-B)$ is a positive (nonnegative) definite matrix for symmetric A and B. $A^{\frac{1}{2}}$ denotes the matrix square root for a nonnegative definite, symmetric A. $\{\Omega, \mathcal{X}, \mathcal{P}\}$ is a probability field. The mean value of a weakly stationary random vector $x(w)$ $w\varepsilon\Omega$ will be represented by $E\{x\}$ or $\overline{x}$ and we will use the abbreviation m.s. for mean square.

MAIN RESULTS

Let us consider the linear discrete-time stochastic-parameter system

$$x_{k+1}=A_k(w)x_k+B_k(w)u_k \tag{1}$$

where $x_k\varepsilon R^n$ is the state, $u_k\varepsilon R^m$ the input vector. x_o is deterministic. The coefficient matrices $A_k(w)$ and $B_k(w)$ have purely random elements which are weakly stationary implying, e.g. $E\{A_k(w)YA_k(w)\}=E\{A^T YA\}$ for an arbitrary constant matrix Y. The elements of the coefficient matrices may be correlated among themselves and with each other, in general.

The above system (or the pair (A,B)) will be called m.s. stabilizable if there exists a constant feedback controller such that the application of that controller in (1) will result in a closed loop system which is m.s. asymptotically stable. A test for m.s. stabilizability is given in [12].

Now, let us also consider the measurement equation

$$y_k=C_k(w)x_k \tag{2}$$

together with the state equation (1). $y_k\varepsilon R^p$ is the output vector and $C_k(w)$ has purely random elements which may be correlated with the elements of $A_k(w)$ and $B_k(w)$. (A,C) will be called m.s. observable if $E\{\|y_k\|^2\}=0$ for all k, implies

$x_o=0$. This definition is shown in [12], to result in the following test for m.s. observability: $\mathcal{A}:R^{nxn}\to R^{nxn}$ defined by $\mathcal{A}(X)=E\{A^TXA\}$, $A_k(w)$ given by (1), needs to satisfy

$$\sum_{k=o}^{n(n+1)/2} \mathcal{A}^k(\overline{C^TC})>0.$$

In the works [14], [17], the following moving horizon performance index

$$E\{J\}=E\{x_{k+N}^TFx_{k+N}+\sum_{i=k}^{k+N-1} x_i{}^TQx_i+u_i{}^TRu_i\} \tag{3}$$

for symmetric and positive definite weighting matrices F, Q and R, is minimized subject to system (1) and the following controller is obtained:

$$u_k= -(R+\overline{B^TP_{N-1}B})^{-1}\ \overline{B^TP_{N-1}A}x_k=K(P_{N-1})x_k \tag{4}$$

where P_{N-1} is found by the (N-1)-fold application of the transformation Φ: $R^{nxn}\to R^{nxn}$, on F so that:

$$P_{N-1}=\Phi^{N-1}(F) \tag{5}$$

with

$$\begin{aligned}\Phi(Y)&=\overline{A^TYA}-\ \overline{A^TYB}(R+\overline{B^TYB})^{-1}\overline{B^TYA}+Q \\ &=E\{[A+BK(Y)]^TY[A+BK(Y)]\}+K^T(Y)RK(Y)+Q\end{aligned} \tag{6}$$

For this constant feedback controller to be stabilizing in both m.s. and almost surely asymptotic senses, it is required that

$$F\geq\Phi(F) \tag{7}$$

In the first main result given below, we will show that for m.s. observable systems, (7) is equivalent to the m.s. stabilizability of system (1).

Theorem 1. Consider matrix inequality (7), with the symmetric weighting matrices satisfying $F,Q\geq0$ and $R>0$. Then:

(a) If system (1) is m.s. stabilizable, then there exist weighting matrices such that (7) holds.

(b) If (7) holds and $(A,Q^{\frac{1}{2}})$ is m.s. observable then the system (1) is m.s. stabilizable and $F>0$.

The proof is made by using some results on generalized Lyapunov equations and the stochastic version of a result in [18].

Next, with the relaxed condition of m.s. observability of $(A,Q^{\frac{1}{2}})$, the stabilizing property of the moving horizon controller will be proved.

Theorem 2. Let (7) hold and $(A,Q^{\frac{1}{2}})$ be m.s. observable. Then the moving horizon control (4) for any $N \geq 1$, is m.s. stabilizing for system (1).

The proof depends on the monotonicity property of the transformation given by (6).

Having proved the m.s. stabilizability of the moving horizon controller under milder conditions, we turn to the infinite horizon case.

Theorem 3. Let (7) hold and $(A,Q^{\frac{1}{2}})$ be observable, then $P=\lim_{k\to\infty} \Phi^k(F)$ exists and is the unique symmetric positive definite solution of the algebraic matrix equation $P=\Phi(P)$. This steady state solution when used in (4) as

$$u_k^s=K(P)x_k$$

is also m.s. stabilizing for system (1).

The existence of the unique limit P depends on the monotonicity property. Stabilizability is shown in the same way as in theorem 2.

The m.s. stabilizability of the steady state solution $P=\Phi(P)$ of (5) starting with any $F \geq 0$ was proved for the first time in [12]. In that work, m.s. stabilizability of (A,B) and m.s. observability of $(A,Q^{\frac{1}{2}})$ are used as the conditions. Since our condition (7) is equivalent to m.s. stabilizability of (A,B) for systems and cost criteria with m.s. observable $(A,Q^{\frac{1}{2}})$, the above result could also be proved using these facts and the results in [12], but, we thought it would be instructive to cbtain the steady state results as natural extentions of finite-time results.

CONCLUSION

In this work, we have shown that a matrix inequality used for moving horizon controllers designed for linear discrete-time stochastic-parameter systems is equivalent to the mean square stabilizability of this class of systems, if the systems in question satisfy a mean square observability condition. This result implies, under proper mean square stabilizability and observability conditions, that a stochastic-parameter system can be mean square stabilized not only by the positive definite solution of a Riccati-like algebraic equation which results from the use an infinite-horizon performance index, but any finite-time solution of the Riccati-like difference equation which results from a moving horizon performance index.

REFERENCES

[1] W.L. Levinson, D.L. Kleinman, and S. Baron, Tech.Rep.1731, Bolt, Beranek and Newman, Inc., Cambridge, MA, 1968.

[2] P.J. McLane, IEEE Trans. Aut. Control, AC-16, 793, 1971.

[3] M. Aoki, IEEE Trans. Aut. Control, AC-20, 388, 1975.

[4] H.J. Kushner, and L. Tobias, IEEE Trans. Aut. Control, AC-14, 219, 1969.

[5] R.G. Agniel, and E.I. Jury, SIAM J. Control, 9, 372, 1971.

[6] W.L. DeKoning, Int.J.Syst.Sci., 11, 841, 1980.

[7] A.J.M. Van Wingerden, and W.L. DeKoning, IEEE Trans.Aut.Control, AC-29, 385, 1984.

[8] E. Yaz, Int. J. Syst. Sci., 17, 819, 1986.

[9] —— , Int. J. Control, 44, 203, 1986.

[10] R.E. Kalman, Proc. 13th Symp. Appl. Math, 13, 287, 1962.

[11] R. Ku and M. Athans, IEEE Trans.Aut.Control, AC-22, 866, 1977.

[12] W.L. DeKoning, Automatica, 18, 443, 1982.

[13] J.L. Willems and J.C. Willems, Automatica, 12, 277, 1976.

[14] E. Yaz, Int. J. Control, 42, 33, 1985.

[15] —— , IEEE Trans.Aut.Control, AC-31, 178, 1986.

[16] —— , ibid, 586, 1986.

[17] —— , Opt. Control Applic. Methods, 8, 327, 1987.

[18] G.A. Hewer, IEEE Trans. Aut. Control, AC-16, 382, 1971.

DRIFT TRANSFORMATIONS USING FEEDBACK CONTROLS AND APPLICATIONS TO SMALL-NOISE PROBLEMS

EFTHIMIOS KAPPOS
Department of Electrical and Computer Engineering,
Northeastern University, Boston, Mass. 02115.

1. INTRODUCTION

This work continues the study of stochastic systems from a dynamical systems point of view initiated in [3],[4],[5]. The relevance of this approach is most evident in the context of small-noise problems, as in [4],[5]. It has been argued there that the most natural way to obtain the small-noise asymptotics is through the associated control system and an optimal control problem related to it. Also, it was seen that several qualitative features of exit paths can be found when the drift vector field is dissipative.

Here we examine in detail drift transformations that use a global feedback control law in the change of measure. In the small noise context, this eliminates the need for a Legendre transformation (see [1]) and leads to a simpler formulation of the results (see [4], [5].)

2. DRIFT TRANSFORMATIONS AND SUPPORT THEOREMS

The transformation of drift (*Girsanov or Cameron-Martin transformation*) has been central to the development of stochastic system theory. It has found applications in many areas such as nonlinear filtering and stochastic optimal control. In the theory of large deviations it is used in the proof of the lower bound for the asymptotic probability of the diffusion following a pre-described path ([1], chapter 3, pp.74–77.)

Suppose the solution x_t of the stochastic differential equation

$$dx_t = b(x_t)dt + \sigma(x_t)dw_t \tag{1}$$

is given on the probability space $(\Omega, \mathcal{F}, P)$ and is adapted to the filtration $\mathcal{F}_t$, relative to the $\mathcal{F}_t$-Brownian motion w_t (throughout, $t \geq 0$.) Here, b is a smooth vector field in $\mathbf{R}^n$ (the **drift vector field**) and σ is a smooth map from $\mathbf{R}^n$ to the space of $n \times m$ matrices (the **diffusion matrix**) and they are such that the solutions exist *globally*, i.e. the explosion time is a.s. infinite (for example, a growth condition such as global Lipschitzness is satisfied.)

We are interested in taking $\mathbf{X}_t$ of the form $\mathbf{X}_t = u(x_t)$, where $u : \mathbf{R}^n \to \mathbf{R}^m$ is a smooth **feedback control law** and x_t is the solution of (1) in the change of measure

$$M_t = \exp[\mathbf{X}_t - \langle \mathbf{X} \rangle_t / 2]. \tag{2}$$

THEOREM: (Transformation of Drift, [2]) *If u is bounded or if it satisfies the condition*

$$E\Big[\exp\big(\frac{1}{2}\int_0^t |u(x_s)|^2 ds\big)\Big] < \infty, \quad \forall t > 0, \tag{3}$$

then

$$M_t = \exp\big(\int_0^t u(x_s)dw_s - \frac{1}{2}\int_0^t |u(x_s)|^2 ds\big) \tag{4}$$

is an $\mathcal{F}_t$-martingale. Define the measure $\tilde{P}$ by $M_t = \frac{d\tilde{P}}{dP}|_{\mathcal{F}_t}$. This is a probability measure, mutually absolutely continuous with respect to P ($\tilde{P} \ll P$ and $P \ll \tilde{P}$.) Also

$$\tilde{w}_t = w_t - \int_0^t u(x_s)ds \tag{5}$$

is a $\mathcal{F}_t$-Brownian motion on the probability space $(\Omega, \mathcal{F}, \tilde{P})$ and is such that $(x_t, \tilde{w}_t)$ is the solution of the stochastic differential equation

$$dx_t = [b(x_t) + \sigma(x_t)u(x_t)]dt + \sigma(x_t)d\tilde{w}_t \tag{6}$$

on $(\Omega, \mathcal{F}, \tilde{P})$ *with filtration* $\mathcal{F}_t$, $t \geq 0$.

In the small-noise context, we need to consider the *stopped diffusion*, obtained as follows: given a domain $\mathcal{D}$ in $\mathbf{R}^n$ such that the *first exit time* τ from $\mathcal{D}$ is a.s. finite, define the process $x_{t\wedge\tau}$ and the feedback $u(x_{t\wedge\tau})$.

COROLLARY: *The conclusions of the theorem above remain true when we consider the stopped process on the stochastic interval* $[0,\tau]$, *under the condition that the feedback control* u *defined in the domain* $\mathcal{D}$ *is bounded (with the necessary modifications in the statements of the results.)*

Support Theorem

THEOREM: *The supports of the diffusions (1) and (6) coincide, i.e.*

$$\mathcal{S}(P_x) = \mathcal{S}(\tilde{P}_x). \tag{7}$$

We have the following result, relating the controllability properties of the control system

$$\dot{x}_t = b(x_t) + \sigma(x_t)u(x_t) \tag{8}$$

to the support in state space of the diffusion and its Markov transition probability

THEOREM: *Suppose that for* K *an attractor of* b *and for all* $t > 0$, *the set of accessible points from a* $x \in R_K$ *of the control system (8) is the whole of* R_K. *Then, the transition probability* $P(t,x,A)$, *for* $A \subset R_K$ *defined in* R_K *has a smooth density* $p(t,x,y)$.

3. FEEDBACK CONTROLS, OPTIMALITY AND APPLICATIONS

LEMMA: *If the smooth feedback control* $u : \mathbf{R}^n \to \mathbf{R}^m$ *is given, there is a smooth map* $p : \mathbf{R}^n \to \mathbf{R}^n$ *such that* u *can be written as* $u(x) = \sigma^T(x)p(x)$.

The interest in the above lemma stems from the fact that the **optimal feedback control** solving an optimization problem *naturally* associated to the diffusion is of exactly this form. It turns out that for the small-noise problem, when the evaluation of the probability of **large deviation trajectories** is of interest, the relevant problem is the **optimal control problem** consisting of (8) and the cost functional

$$J(u) = \frac{1}{2}\int_0^T |u_s|^2 ds. \tag{9}$$

Solving this problem leads to the **optimal feedback law** (where p is the dual variable of the Hamilton flow)

$$u^*(x) = \sigma^T(x)p(x) \tag{10}$$

under some conditions of weak controllability—see the definition of **Lyapunov controllability** in [5]. This condition requires the law (10) to be non-zero in R_K, i.e. $\sigma^T(x)p(x) \neq 0$.

REFERENCES

[1] M.I. Freidlin, A.D. Wentzell: *Random Perturbations of Dynamical Systems*, Springer, 1984.

[2] N. Ikeda, S. Watanabe: *Stochastic Differential Equations and Diffusion Processes*, North–Holland 1981.

[3] E. Kappos: *'A global optimal control setting for large deviations'*, 25th Allerton Conference, 1987.

[4] E. Kappos: *'Large deviations for singular diffusions with dissipative drift'*, 1988 Conf. on Information Sciences and Systems, Princeton.

[5] E. Kappos, S. Sastry: *'Lyapunov controllability and global optimal control'*, 25th Control and Desicion Conference, Athens, 1986.

A SPLITTING ALGORITHM FOR COMMUNICATION WITH REAL-TIME DELIVERY CONSTRAINTS

DAN C. MARINESCU[*]
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

Abstract

In this paper, we present a random multiple access algorithm with real-time delivery constraints (RTD). RTD is a non-blocking splitting algorithm with splitting based upon a modified extinction time.

1. INTRODUCTION

In this paper, we propose an algorithm for random multiple access, when messages have hard real-time constraints.

Communication with hard real-time constraints has applications in data transmission, as well as in integrated systems, with a mixed traffic consisting of voice, images and data. Typical applications of data transmission with hard real-time constraints, are distributed systems operating in a local area network environment, used for process control, automated manufacturing, surveillance, etc. Such systems can be modeled as a set of nodes, interconnected by a shared communication channel. The problem investigated in this paper is to design a distributed scheduling algorithm, such that each node is capable of scheduling the transmission of messages generated locally observing the real-time constraints.

Several classes of multiple access methods exist. *Collision-free* methods are usually based upon a token passing scheme. A node transmits when it receives the permission (the token or the virtual token). A protocol, FDDI, which allows a mix of synchronous traffic (traffic with real time constraints, voice or images), and asynchronous traffic (data) is in a standardization phase. In FDDI the communication bandwidth is statically allocated among the nodes to satisfy the requirements of the synchronous traffic. The asynchronous traffic is controlled by means of a timed token. If the token arrives early at a given node, then the node is allowed to send asynchronous traffic, and the amount of data is determined by the earliness of the token arrival.

There are several classes of collision based methods for random multiple access. Aloha, carrier sense methods and splitting algorithms are notable examples. While Aloha is mostly of theoretical interest, CSMA is widely used in local area networks. Several splitting algorithms, also known as *Contention Resolution Algorithms*, have been investigated thoroughly. Among these, we note the algorithm of Capetanakis, the CTM stack algorithm [3], the FCFS algorithm of Gallager [1], [4], [5]. But none of the traditional random multiple access algorithms is suitable for real-time communication. Any algorithm which schedules the transmission of messages without taking into account their deadline cannot guarantee the delivery of messages with real-time constraints.

---

[*] Work supported in part by NSF grant NCR-8702115.

Recently, the interest in random multiple access algorithms with hard real-time constraints, seems to be growing. Zhao and Ramamritham [8] have proposed a Virtual Time CSMA Protocol, and Zhao, Stankovic and Ramamritham [9] have discussed a multiaccess window protocol. Kurose, Schwartz and Yemini [6] have studied window protocols.

Let us now outline the basic concepts and terminology for communication with hard real-time constraints. A message upon its arrival at time t_i makes known its *deadline* d_i. If the transmission of the message does not begin by time $e_i = t_i + d_i$, the message is considered lost. We follow the terminology discussed in [2] and call e_i the *extinction* time of the message.

The algorithm we propose, belongs to the class of non-blocking splitting algorithms with splitting determined by the *modified extinction time.* The modified extinction time e'_i of a message with extinction time e_i, is a random variable uniformly distributed in the interval $\min([e_i - \delta, e_i], [e_i - t_i + \Delta, e_i])$ with δ, Δ parameters of the algorithm.

Our algorithm requires that the deadline d_i of any message accepted satisfies the condition

$$d_i \geq \Delta$$

with Δ called the *minimum acceptable deadline*. It seems rather obvious that when $\Delta \to 0$ any message scheduling policy will eventually fail, since some messages will have the extinction time equal to the current time. On the other hand, when Δ is large the algorithm does not accept a number of messages, though scheduling strategies which guarantee the delivery of all messages may exist. We recognize two kinds of losses in our system: messages which are not accepted, since their deadline is too short and messages lost, since their extinction time has passed. Given the distribution of the message arrival time, the arrival rate and the distribution of the delivery time, an optimal value for Δ is the one which minimizes in some sense the total number of lost messages.

2. THE ALGORITHM

The basic idea of splitting algorithms is to divide (to split) successively the group of nodes which are allowed to transmit in an interval until the group has a size of at most one, and a successful transmission is guaranteed. The splitting can be done using some characteristics of a node, for example, the Capetanakis algorithm uses the node identification (all nodes form the leaves of a tree). Other algorithms split the nodes probabilistically. In the CTM algorithm each node involved in a collision flips a coin and the splitting is determined by the outcome of the coin flipping. Finally, there are algorithms like the FCFS algorithm of Gallager which allow each node to compute a window and any node with a packet having the arrival time within the window is allowed to transmit. The window is then successively split, until it contains at most one packet. The algorithm guarantees the transmission of the packets in the order of their arrival. The FCFS algorithm is described in [1] and the rules to compute the position, and the size of the window, as well as the state of the system are given. The state determines which segment of the window, L = left, R = right is allowed to transmit.

Clearly, the splitting can be based upon the requested packet delivery time. A node is allowed to transmit if the extinction time of a packet falls within a window known to all nodes. In this case the distribution of the extinction time is determined by the arrival process and by the distribution of the deadlines. Unfortunately, while it is fairly reasonable to assume a Poisson distribution of the packet arrival times, the distribution of the packet extinction time may well not be Poisson. As a result, the probability of having two packets with the same extinction time is not zero, and we have to provide a solution for such "ties". Consequently our algorithm uses a modified extinction time rather than the extinction time for scheduling the transmission of a packet.

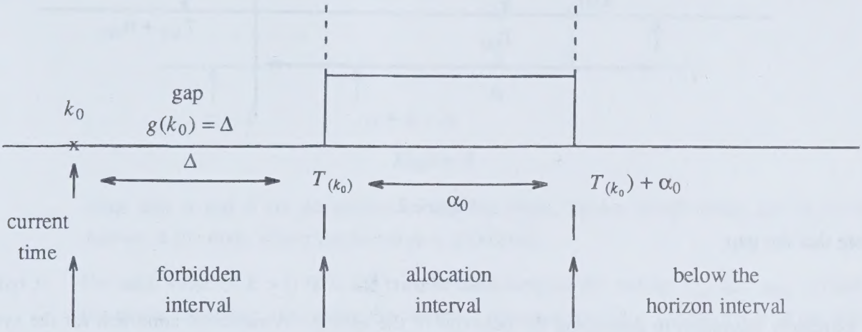


Figure 1.

The basic idea of the algorithms to partition the time scale into three regions

(a) a forbidden interval, containing packets which cannot be accepted for transmission,

(b) an allocation interval, containing packets which are currently being scheduled,

(c) below the horizon interval, containing packets to be scheduled for transmission sometime in the future.

From Figure 1 we observe that the allocation interval is now located in the future as opposed to the 0.487 algorithm. The real-time delivery constraints impose that **the gap**, the distance between the left hand side of the allocation interval and the current time, be positive at all times.

Let us now describe informally the algorithm. Each node with a message (t_i, e_i) computes its modified extinction time e_i'. Note that in order to simplify the description of the algorithm we consider a time slotted channel and t_i, e_i, and e_i' will be the slots where the corresponding events have occurred (t_i, the slot when the packet was generated, e_i the extinction slot, e_i', the modified extinction slot).

Let us now start the system such that the current time is slot k_0. From Figure 1 we see that initially the gap between the left hand side of the allocation interval and the current time $g(k_0) = T(k_0) - k_0$ is equal to the minimum acceptable deadline Δ, and that the size of the allocation interval α_0 is determined from performance optimization criteria. If we call λ the cumulative arrival rate in the system then α_0 is selected such that the length of collision resolution interval with $N = \lambda \times \alpha_0$ packets is smaller than Δ.

In the next slot each node will compute $T(k_0 + 1)$ and $\alpha(k_0 + 1)$ function of the feedback from the channel (collision, idle, success). At a later moment of time, say $k = k_0 + i$, the relative position of the three regions may be the one presented in Figure 2.

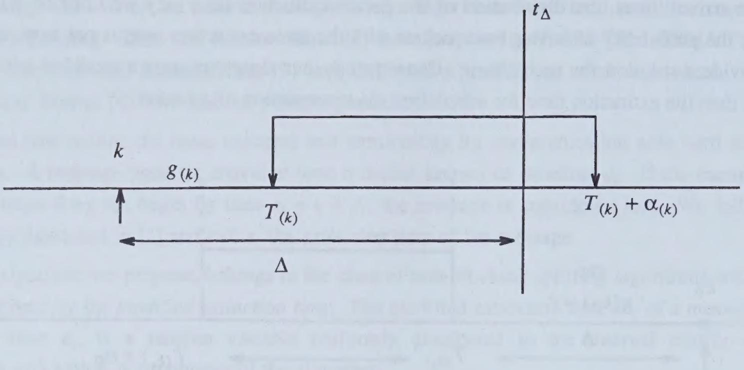


Figure 2.

Note that the gap

$$g_{(k)} = T_{(k)} - k$$

is extremely important in describing the behavior of the system. A sufficient condition for the system to meet all the deadlines is that

$$g_{(k)} \geq 1 \text{ for all } k.$$

Note also that this condition might not be necessary, since though the modified extinction time of a packet may have passed, it is still possible that its true extinction time e_i is in the future. Also the gap is not allowed to be larger than the minimum acceptable deadline

$$g_{(k)} \leq \Delta.$$

Now we give the formal description of a first RTD (Real-Time Delivery) splitting algorithm.

Step 1: Given a packet (t_i, d_i) a node computes its extinction time $e_i = t_i + d_i$. It drops the packet if

$$d_i < \Delta.$$

Step 2: If $d_i \geq \Delta$, then the node computes the modified extinction time e_i' as a random number uniformly distributed in the interval I defined by

$$I = \min([e_i - \delta, e_i],\ [e_i - t_i + \Delta, e_i])$$

as shown in Figure 3.

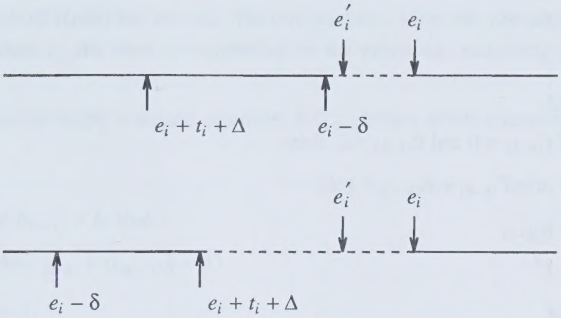


Figure 3.

Note that Δ and δ are parameters of the algorithm, known to all nodes and (t_i, e_i) are known at the node where the message is generated.

Step 3: For each value of $k > 0$ (k is the current slot) compute the values $T_{(k)}$ and $\alpha_{(k)}$ according to the following rules using the feedback information $f(k-1)$, the status of the system $\sigma(k-1)$ and the previous values of of the left hand side margin of the allocation interval, $T_{(k-1)}$, and of the size of the allocation interval, $\alpha_{(k-1)}$. Note that initially $T_{(0)} = \Delta, \quad \alpha_{(0)} = \alpha_0, \quad \sigma_{(0)} = R$.

The following convention is used to describe the feedback information: c means collision, 1 means successful transmission and 0 means an idle slot.

Rule 3.1.

If $\alpha_{(k-1)} \le \alpha_{\min}$, then all nodes involved in a collision compute a new modified extinction time e_i'' as a random number uniformly distributed over the interval $I' = \min([e_i' - \delta, e_i'], [e_i' - t + \Delta, e_i'])$. All nodes use the same values for $T_{(k)}$, $\alpha_{(k)}$, $\sigma_{(k)}$, as in the previous slot namely

$$T_{(k)} = T_{(k-1)}$$

$$\alpha_{(k)} = \alpha_{(k-1)}$$

$$\sigma_{(k)} = L$$

Note that for a time slotted channel $\alpha_{\min} = 1$.

If $\alpha_{(k-1)} > \alpha_{\min}$, then all nodes compute the new values for $T_{(k)}$, $\alpha_{(k)}$, $\sigma_{(k)}$, as follows

$$T_{(k)} = T_{(k-1)}$$

$$\alpha_{(k)} = \frac{\alpha_{(k-1)}}{2}$$

$$\sigma_{(k)} = L$$

Rule 3.2.

If $f_{(k-1)} = 1$ and $\sigma_{(k-1)} = L$, then

$$T_{(k)} = \min[(T_{(k-1)} + \alpha_{(k-1)}), (k + \Delta + 1)]$$

$\alpha_{(k)} = \alpha_0$

$\sigma_{(k)} = R$

Rule 3.3.

If $f_{(k-1)} = 0$ and $\sigma_{(k-1)} = L$,then

$T_{(k)} = \min(T_{(k-1)} + \alpha_{(k-1)}, k + \Delta)$

$\alpha_{(k)} = \alpha_{(k-1)}$

$\sigma_{(k)} = L$

Rule 3.4.

If $f_{(k-1)} = 0$ or 1 and $\sigma_{(k-1)} = R$, then

$T_{(k)} = \min(T_{(k-1)} + \alpha_{(k-1)}, k + \Delta)$

$\alpha_{(k)} = \alpha_0$

$\sigma_{(k)} = R$

Rule 4. Transmit a packet with the modified extinction time e_i' in slot k iff

$$T_{(k)} \le e_i' < T_{(k)} + \alpha_{(k)}$$

Discharge a packet if

$$e_i' < T_{(k)}$$

It is important to observe that the number of packets in an allocation interval of a given size may increase from one slot to the next due to new packet arrivals. This problem does not occur in other splitting algorithms since the allocation interval is located in the past, it contains all the packets the system has to deal with. In our case the allocation interval is in the future, and new packets which satisfy the minimum deadline requirements must be accepted. For this reason, we cannot push the left hand side of our allocation interval further than Δ.

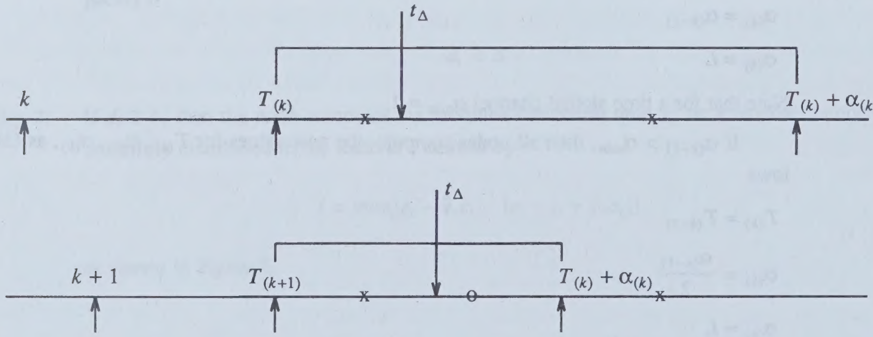


Figure 4.

Figure 4 illustrates such a situation. In slot k we have a collision (there are two packets in the allocation interval, marked as ×). The collision would have been solved in slot $(k + 1)$, but a new packet (marked as a small circle) has arrived. The new packet is accepted, since its modified extinction time is larger than t_Δ the time corresponding to the minimum acceptable deadline for slot $k+1$.

Let us now describe briefly a second algorithm RTD2. It uses another version of rule 3.2. The new rule is:

Rule 3.2.

If $f_{(k-1)} = 1$ and $\sigma_{(k-1)} = L$, then

$$T_{(k)} = \min(T_{(k-1)} + \alpha_{(k-1)}, k + \Delta)$$

$$\alpha_{(k)} = \alpha_{(k-1)}$$

$$\sigma_{(k)} = R$$

Note that in both algorithms when an idle slot follows a collision, the right most interval contains a collision and it is immediately split (Rule 3.3). Note also that when a collision follows another collision, the interval on the right hand side of the second collision is returned to the below the horizon interval. The remaining algorithms of the family do not return the right hand side interval of the second collision to the below the horizon interval.

3. SCHEDULING IN DISTRIBUTED REAL-TIME SYSTEMS AND PERFORMANCE CRITERIA

Scheduling in a system with real-time constraints raises fairly difficult problems, especially when non-determinism is taken into account, for example when the exact arrival time of requests is not known, only the distribution of the arrival time is given, and only service times distribution is available. Not only it is difficult to study scheduling policies in such a real-time system, but even the performance measures for the system have to be investigated from a slightly different perspective.

The issues become even more difficult in a distributed system, when the global state of the system is not known to a central scheduler and a distributed decision making process takes place. Individual nodes take actions based upon partial knowledge concerning the system state. This is precisely the case of random multiple access when each node has to schedule the transmission knowing only its own deadline.

It seems obvious that scheduling policies which assign higher priority to tasks with the nearest deadline are good candidates for optimal scheduling policies. Ephremides and Bhattacharaya [1] have shown that scheduling policies based upon STE (Shortest-Time to Extinction) are optimal in the class of non-idling, non preemptive policies for hard real-time systems. The optimality of a scheduling policy is related to the minimization, in the sense of stochastic order of the number of messages lost over any interval.

A splitting algorithm which would schedule the transmission of a packet such that the packets with the shortest time to extinction are transmitted first would be optimal. We are forced to use a PSTE (Pseudo-STE) scheduling policy to avoid possible ties caused by an STE scheduling. The algorithm fails if two nodes attempt to transmit two messages with precisely the same extinction time, $e_i = e_j$. For this reason, we follow an idea of Jacquet [8] to redistribute message extinction time over an interval with the upper limit equal to the actual extinction time.

Different classes of random multiple access methods have been investigated [3], [4], [5], etc. The analysis of such algorithms is focused upon stability conditions, as well as upon delay analysis. To compare two random multiple access algorithms we study the largest arrival rate for which the algorithms are stable and the delay characteristics of the two algorithms.

Analysis of algorithms for hard real-time communication adds new twists to the analysis, namely, when should we say that such an algorithm is stable, and what measures of performance are significant in this case. Clearly the objective is to minimize, in some sense, losses. Delay analysis is significant only to the extent that it provides answers to the question if the deadline is met or not, but the actual waiting time is not of interest.

Stability in case of systems with hard real-time constraints seems difficult to define. Several alternatives can be considered. For example we may say that a system is stable if the expected number of packets lost satisfies the condition

$$E(L) < \varepsilon \quad \text{for a given } \varepsilon.$$

It seems acceptable to define a penalty function $P(m_i)$ associated with message m_i and say that the system is stable if the cumulative penalty associated with the loss of messages $m_i \in Q$ satisfies the condition

$$\sum_{i \in Q} P(m_i) < \varepsilon_P \quad \text{for some } \varepsilon_P.$$

We may also ask whether such performance measures should be defined in the sense of stochastic order instead of defining them in terms of averages.

Another approach is to consider the loss of a packet as a failure of the system and study the first passage time to the group of system states where failures occur. As a first approximation, we study only losses of the second type (the extinction time has passed) and ignore losses of the first type (deadline too short). In the case of RTD algorithms, in continuous time, we can probably model the system as a two dimensional Markov chain $(g_{(t)}, \alpha_{(t)})$, the state of the system as a given time t being determined by the gap, $g_{(t)}$ and the size of the allocation interval, $\alpha_{(t)}$.

In this case, let us call F the group of states when deadlines are not met. F consists of all states when $g_{(t)} = 0$. The first moment of the passage time to F is another possible measure used to compare two algorithms.

4. ON THE MODELING AND ANALYSIS OF THE RTD ALGORITHMS

In this section, we discuss briefly two possible approaches to the analysis [7] of the RTD algorithm.

A first approach is to exploit the analogy of the RTD and FCFS algorithms and model the system as a Markov or a semi-Markov process, in which the system state at time $t = k$ is given by the tuple $(\sigma_{(k)}, i_{(k)})$ with $\sigma_{(k)} = L, R$ the current sub-interval and $i_{(k)}$ the level of splitting. The corresponding chain is shown in Figure 5 (for details see [1]).

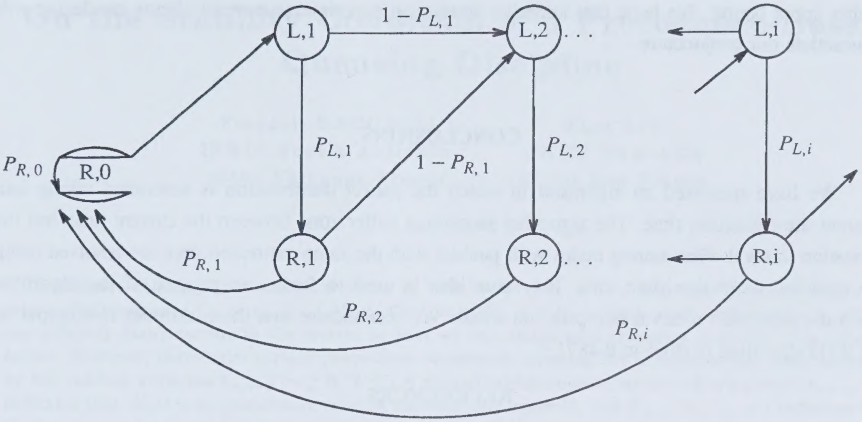


Figure 5.

In this case, the top states correspond to the left intervals. The state (L,i) $i > 1$, is entered

- when a collision in state $(L,i-1)$ occurs, or
- when a collision in state $(R,i-1)$ occurs, or
- when an idle slot in state $(L,i-1)$ occurs.

The state (R,i) is entered in case of a success in state (L,i). A successor or an idle slot as state (R,i) marks the termination of a collision resolution interval and a transition to the initial state $(R, 0)$.

If K is the number of slots in a collision resolution interval with k packets we have to express $E\ (K)$ in terms of $P_{L,i}$ and $P_{R,i}$, see Figure 5. Then a condition for real-time delivery is

$$E\ (K) < \Delta.$$

A second approach used successfully for the delay analysis of other splitting algorithms is based upon identification of regeneration points in the system, [4]. Such points can be easily identified in case of the 0.487 algorithm as the points where the lag (the distance between the right hand side margin of the allocation interval and the current time) is one. These are the moments when the system is empty, all packets have been successfully transmitted. Unfortunately the identification of such regeneration points could prove to be difficult for the RTD algorithm. In this case it is not possible to detect when the system is empty from the size of the gap, or the position and size of the allocation interval. The system does not know anything about packets with extinction time below the horizon, and, even more drastic, some of these packets may not have been generated yet.

As far as the maximum throughput of RTD algorithms is concerned it is clear that

$$\lambda < 0.4871.$$

We conjuncture that for optimal values of Δ, δ, α_0 the maximum throughput of the algorithm is close to this upper bound. We hope that intensive simulation experiments currently being conducted will substantiate our conjuncture.

CONCLUSIONS

We have presented an algorithm in which the packet transmission is scheduled taking into account the extinction time. The algorithm requires a buffer zone between the current time and the allocation interval. Ties among nodes with packets with the same extinction time are resolved using the modified extinction time idea. The same idea is used to force convergence of the algorithm when the allocation interval becomes too small. We conjuncture that the maximum throughput of the RTD algorithm is close to 0.4871.

REFERENCES

[1] D. Bertsekas and R. Gallager, "Data Networks", Prentice Hall, 1987.

[2] P. Bhattacharaya and Anthony Ephremides, "Optimal Scheduling with Strict Deadlines", Research Report, University of Maryland, 1988.

[3] Guy Fayolle, P. Flajolet, M. Hofri and P. Jacquet, "Analysis of a Stack Algorithm for Random Multiple Access Communication", IEEE Trans. on Int. Theory, Vol. IT-31, No. 2, pp. 244–544, 1985.

[4] L. Georgiadis, L.F. Merakos and P. Papantoni-Kazakos, "A Method for the Delay Analysis of Random Multiple Access Algorithms Whose Delay Process is Regenerative", IEEE Journal on Selected Areas of Communications, Vol. SAC-5, No. 6, pp. 1051–1062, 1986.

[5] J.-C. Huang and T. Berger, "Delay Analysis of 0.487 Contention Resolution Algorithms", IEEE Trans. on Communications, Vol. COM-34, No. 9, pp. 916–926.

[6] J.F. Kurose, M. Schwartz and T. Yermini, "Controlling Window Protocols for Time Constrained Communication in a Multiple Access Network", IEEE Trans. on Communication, Vol. 36, No. 1, pp. 41–49, January 1988.

[7] D.C. Marinescu and W. Szpankowski, "On the Analysis of RTD Splitting Algorithms", working paper, June 1988.

[8] W. Szpankowski, Private communication.

[9] W. Zhao and K. Ramamritham, "Virtual Time CSMA Protocols for Hard Real-Time Communication", IEEE Trans. on Software Engineering, Vol. SE-13, No. 8, pp. 938–952, August 1987.

[10] W. Zhao, J.A. Stankovic and K. Ramamritham, "A Multiaccess Window Protocol for Transmission of Time Constrained Messages", Proc. 8th Int. Conference on Distributed Computing Systems, pp. 384–392, 1988.

On the Stability Condition of a Precedence-Based Queueing Discipline

François BACCELLI
INRIA-Sophia Antipolis
06565 Valbonne France

Zhen LIU
CNET PAA-ATR
92131 Issy France

July 1988

Customers $C_0, C_1, C_2, \cdots$ arrive at dates $t_0 = 0 < t_1 < t_2 < \cdots$, respectively, which define the inter-arrival sequence $\{\tau_n = t_{n+1} - t_n\}_{n=0}^{\infty}$. Customer C_n, $n \geq 0$, requires service time $\sigma_n > 0$. There are infinitely many servers in the system so that we can assign each incoming customer to a different server. However, there exist certain precedence constraints between the customers which are indicated by the random variables $b_{n,n-j}$ $(n \geq 0,\ 1 \leq j \leq n)$ and which operate in the following way: $b_{n,n-j} = 0$ indicates that there is no precedence relation between customers C_n and C_{n-j}; $b_{n,n-j} = 1$ indicates that C_n has to wait for the end of the service of C_{n-j} before it starts service. This queueing system is a very simple model for database concurrency control via "static" locking, as well as for parallel execution of programs consisting of several interdependent processes.

This precedence-based queueing system was introduced by J.N. Tsitsiklis, C.H. Papadimitriou and P. Humblet in [5]. In the case where the customer service times are deterministic, the arrival process is Poisson and the precedence variables form independent i.i.d. sequences, these authors derived the stability condition and established that this condition is of threshold type on the intensity of the arrival process. For more general statistical assumptions they conjectured that the stability condition would depend on the complete service time distribution functions but only on the first moment of the inter-arrival times.

In this paper, we prove the above conjecture under the following statistical assumptions: the service and inter-arrivals times form jointly stationary and ergodic sequences; the precedence variables are mutually independent and independent of the inter-arrival and service times. Only the main ideas are given together with a sketch of the proof of the main theorem. The interested reader may consult [2] for more details.

Define w_n $(n \geq 0)$ to be the waiting time of customer C_n. In the sequel, the above queueing system will be said to be stable iff w_n converges weakly to a finite Random Variable (RV) w_∞ when n goes to ∞. The evolution of the system is captured by the following set of recursive equations.

Lemma 1 *Assume that the queueing system is empty at time 0. Then,* $w_0 = 0$ *and for all* $n \geq 1$,

$$w_n = \max(0, \max_{1 \leq j \leq n} b_{n,n-j}(w_{n-j} + \sigma_{n-j} - \delta_{n,n-j})) \tag{1}$$

where $\delta_{n,j} = \tau_j + \tau_{j+1} + \cdots + \tau_{n-1}$ *for all* $0 \leq j \leq n-1$. □

The basic idea for determining the stability condition of this queueing system consists in generalizing the schema of Loynes for the waiting times in a $G/G/1$ queue [3]. For this, we will assume without loss of generality that the above sequences are subsequences of three bi-infinite sequences of RV's $\{\tau_n\}_{-\infty}^{\infty}$, $\{\sigma_n\}_{-\infty}^{\infty}$, and $\{b_{n,n-j}, j \geq 1\}_{n=-\infty}^{\infty}$ with values in $I\!R^+$, $I\!R^+$ and $\{0,1\}^\infty$ respectively. These sequences are assumed to be defined on their canonical probability space (Ω, F, P). θ will denote the leftshift operator on this canonical space. For more details on this formalism, see [1]. Within this framework, the above statistical hypothesis read

A_1: *P is θ-invariant and θ-ergodic.*

A_2: *The $\{0,1\}^\infty$-valued sequences $\{b_{n,n-j},\ j \geq 1\}$, $n = \cdots, -1, 0, 1, \cdots$, are mutually independent, and independent of the sequences $\{\tau_n\}_{n=-\infty}^{\infty}$ and $\{\sigma_n\}_{n=-\infty}^{\infty}$; For all $n \in \mathbb{Z}$, the sequence $\{b_{n,n-j}\}_{n=0}^{\infty}$ is made of independent RV's on $\{0,1\}$;*

Observe that this last assumption allows for the probability of the precedence $P[b_{n,n-j} = 1] = p_j$ to depend upon the "distance" between the two customers. In the sequel, it will be assumed that $\inf_{j \geq 1} p_j > 0$.

Let $\tau = \tau_0$, $\sigma = \sigma_0$, $\delta_i = \delta_{0,-i}$ $(i \geq 1)$, and $b_i = b_{0,-i}$ $(i \geq 1)$. Consider the Loynes schema $\{M_n\}_{n=0}^{\infty}$ defined by $M_0 = 0$ and for $n \geq 1$,

$$M_n = \max(0, \max_{1 \leq j \leq n} b_j(M_{n-j} \circ \theta^{-j} + \sigma \circ \theta^{-j} - \delta_j)) \tag{2}$$

Lemma 2 *M_n can be rewritten as $M_n = \max_{1\le m\le n} H_m$, where*

$$H_m \quad = \quad \max_{1\le k\le m} \max(0, \max_{J^k_m} B(J^k_m)\cdot(\sum_{i=1}^{k} \sigma\circ\theta^{-l_i} - \sum_{i=1}^{m} \tau\circ\theta^{-i})) \tag{3}$$

and

$$J^k_m = \{J^k_m = (j_1,\cdots,j_k) \in \mathbb{N}^k | 1 \le j_1,\cdots,j_k \le m, \sum_{i=1}^{k} j_i = m\}$$

$$B(J^k_m) = \prod_{i=1}^{k} b_{j_i}\circ\theta^{-s_i}, \quad s_i = \sum_{h=1}^{i-1} j_h \ , \quad l_i = s_i + j_i$$

with $s_1 = 0$ by definition. □

It can easily be proved by induction that w_n and M_n are equivalent in law (more precisely, we have $w_n = M_n \circ \theta^n$, $n \ge 0$) and that the sequence $\{M_n\}_{n=0}^{\infty}$ is pathwise increasing in n.

Denote by M_∞ the limit of M_n when n goes to ∞. Owing to the foregoing ergodicity assumptions, it can also be shown that the event $\{M_\infty = \infty\}$ is of probability 0 or 1.

Let α be the RV

$$\alpha = \limsup_{n\to\infty} \max_{1\le k\le n} \max_{J^k_n} B(J^k_n)\frac{\sum_{i=1}^{k}\sigma\circ\theta^{-l_i}}{n} \tag{4}$$

Theorem
Assume A_1 and A_2 hold. α is integrable. If $E[\alpha] < E[\tau]$, then $M_\infty < \infty$ a.s., so that w_n converges weakly to a finite RV when n goes to ∞. If $E[\alpha] > E[\tau]$, then $M_\infty = \infty$ a.s. and w_n diverges a.s.. □
Sketch of the Proof.
Owing to the ergodic assumption, $\alpha < E[\sigma]$ a.s., which proves the first assertion.

Assume the event $\{M_\infty = \infty\}$ is of probability 1. Owing to (3), this is equivalent to $\limsup_{n\to\infty} H_n = \infty$, a.s. which in turn implies

$$\limsup_{n\to\infty}(\max_{1\le k\le n} \max_{J^k_n}\{B(J^k_n)\frac{\sum_{i=1}^{k}\sigma\circ\theta^{-l_i}}{n}\} - \frac{\sum_{i=1}^{n}\tau\circ\theta^{-i}}{n}) \ge 0$$

This entails $E[\alpha] \ge E[\tau]$. Taking the contrapositive of this and using the above $0-1$ law, immediately shows that the assumption $E[\alpha] < E[\tau]$ entails $M_\infty < \infty$ a.s. .

Assume now that $E[\alpha] > E[\tau]$. Then there exists a subset $\Omega_0 \subset \Omega$ with $P[\Omega_0] > 0$, such that for all $\omega \in \Omega_0$, $\alpha > E[\tau]$. This and (3) imply $\lim_{n\to\infty} M_\infty = \infty$ with a positive probability and hence a.s.. □
Remark: The Theorem shows explicitly that the stability condition of the precedence-based queueing systems is indeed of threshold type on the intensity of the arrival point process On the other hand, the expression of $E[\alpha]$ defined by (4) is a function of the whole distribution of the service times and precedence variables.

References

[1] F. Baccelli, P. Brémaud, *Palm Probabilities and Stationary queues.* Lecture Notes in Statistics No. 41, Springer Verlag, 1987.

[2] F. Baccelli, Z. Liu, *On the Stability Condition of a Precedence-Based Queueing Discipline.* Rapport INRIA No. 880, July 1988.

[3] R.M. Loynes, *The Stability of Queues with Non Independent Inter-arrival and Service Times.* Proc. Cambridge Ph. Soc. 58, pp .497-520, 1962.

[4] J.N. Tsitsiklis, C.H. Papadimitriou and P. Humblet, *The Performance of a Precedence-Based Queueing Discipline.* J.ACM Vol. 33, No. 3, pp. 593-602, July 1986.

Simple, Efficient Computation of the Generalized Throughput of Jackson Networks

Albert G. Greenberg and William A. Massey
AT&T Bell Laboratories, Murray Hill, NJ

Abstract: A Gauss-Seidel (successive substitutions) method for solving the generalized throughput equation of Jackson networks is considered. This modified equation handles the situation of the total network being unstable, but having a stable subnetwork. Thus for an open network, the solution of this equation identifies the bottlenecks of the system. It is shown that a monotonicity property of the Gauss-Seidel operator can be exploited to produce two sequences of throughput estimates: one converging to the true throughput from below and the other from above. Thus, the two can be computed together and the computation stopped when their difference is small. A simple bound on the rate of convergence is given, and a heuristic for organizing the computation to accelerate convergence is suggested.

1. Solving the Generalized Throughput Equation

The Jackson network [J] is an N node network where the i-th node is an M/M/1 queue with arrival rate $\lambda_i \geq 0$ and service rate $\mu_i > 0$, with $\lambda_i < \mu_i$. The queues are connected by an $N \times N$ switching matrix $\mathbf{P}$, where a customer, after completing service at node i, leaves and joins node j with probability p_{ij}. With probability $q_i = 1 - \sum_{j=1}^{N} p_{ij}$ however, it may decide to leave the network entirely. We assume that $p_{ii} = 0$ for all i. We say a given node j can be *filled* if jobs can reach it: either $\lambda_j > 0$ or there is a node i and a positive integer m such that $p_{ij}^m > 0$, where p_{ij}^m is the (i,j) entry of $\mathbf{P}^m$, and $\lambda_i > 0$. Similarly, we say node j can be *drained* if either $q_j > 0$ or there is a node i and a positive integer m such that $p_{ji}^m > 0$ and $q_i > 0$.

What makes Jackson networks useful in modeling of computer and communication networks is that, in contrast to other perhaps more realistic models, the analysis of Jackson networks is computationally tractable. In this paper, we propose and analyze a simple numerical method for solving the generalized throughput equation for Jackson networks. Having solved that equation we can deduce which nodes are bottlenecks or unstable, and which ones are stable and have a steady state distribution, provided the network is *open*, meaning every node can be filled. Open Jackson networks are often used in approximation schemes for the analysis of more complicated closed networks (see Whitt [W]).

In this paper, the throughput θ_i of a given node i is the net input rate to node i: the external arrival rate λ_i plus the arrival rate of jobs routed to node i from other nodes. Let $Q_i(t)$ denote the queue length at node i at time t. Goodman and Massey [GM] proved the following theorem, generalizing a fundamental result of Jackson [J]. The notation $a \wedge b$ denotes min $\{a,b\}$.

Theorem 0. *Given an N node Jackson network where every node can be filled or drained, there is a unique solution to the following generalized throughput equation*

$$\boldsymbol{\theta} = (\boldsymbol{\theta} \wedge \boldsymbol{\mu})\mathbf{P} + \boldsymbol{\lambda}. \tag{1}$$

Now let $\rho_i = \theta_i / \mu_i$, *and* $S = \{i \mid \theta_i < \mu_i\}$. *If every node can be filled, then* $\lim_{t \to \infty} Pr\,\{Q_j(t) = n_j\,;\, j \in S\} = \prod_{i \in S} (1 - \rho_i)\,\rho_i^{n_i}$, *for all integers* $n_i \geq 0$ *with* $i \in S$. *Moreover, S is the maximal set of stable nodes in that if* $j \notin S$ *then* $\lim_{t \to \infty} Pr\,\{Q_j(t) = n\} = 0$, *for all integers* $n \geq 0$.

The main difficulty in solving (1) is dealing with the non-linearity that the min operator introduces. The unique solution to (1) can be found by using Jacobi iteration as shown in Massey [M]. Goodman and Massey [GM] presented an N pass method where each pass involves solving an $N \times N$ linear system that agrees with (1) except that the min operator is dropped and either μ_i or θ_i is substituted for each term $\theta_i \wedge \mu_i$.

We propose a different way to solve (1) here, based on the Gauss-Seidel method, see Tijms [T] or Varga [V], which is an iterative numerical method. Our method is especially appealing when the routing matrix $\mathbf{P}$ is large and sparse, since it preserves sparsity. At each iteration, a new estimate of the throughput $\boldsymbol{\theta}$ is computed by successive substitution into (1). A common failing of such methods is that the only reasonable rule for stopping the iteration is to stop when the distance between successive iterates is small. This can be dangerous; successive iterates can be close with each still far from the true answer. We show that the Gauss-Seidel operator associated with (1) has a monotonicity property that allows us to produce two sequences of throughput estimates: one converging to the true throughput from below and the other from above. Thus, the two can be computed together and the computation stopped safely when their difference is small.

Consider the generalized throughput equation for a Jackson network (1). Nodes of the network are indexed 1, 2, ..., N, and so they are implicitly ordered. Define the matrix $\mathbf{U}$ to be the strictly upper triangular submatrix of $\mathbf{P}$, or $u_{ij} = p_{ij}$ if $i < j$, and $u_{ij} = 0$ otherwise; and set $\mathbf{L} = \mathbf{P} - \mathbf{U}$. The *Gauss-Seidel operator G* maps R^N to R^N as follows

$$G(\mathbf{x}) = (G(\mathbf{x}) \wedge \boldsymbol{\mu})\mathbf{U} + (\mathbf{x} \wedge \boldsymbol{\mu})\mathbf{L} + \boldsymbol{\lambda}.$$

Our algorithm is to compute two sequences of Gauss-Seidel iterates, $\boldsymbol{\theta}^-(t)$ and $\boldsymbol{\theta}^+(t)$, stopping at the first iteration t where the distance between $\boldsymbol{\theta}^-(t)$ and $\boldsymbol{\theta}^+(t)$ is small. Specifically, let

$$\boldsymbol{\theta}^-(0) = \boldsymbol{\lambda}; \qquad \boldsymbol{\theta}^-(t+1) = G(\boldsymbol{\theta}^-(t)); \; t \geq 0$$

$$\boldsymbol{\theta}^+(0) = \boldsymbol{\mu}\,\mathbf{P} + \boldsymbol{\lambda}; \qquad \boldsymbol{\theta}^+(t+1) = G(\boldsymbol{\theta}^+(t)); \; t \geq 0$$

It is not hard to see that $\boldsymbol{\theta}^-(0) = \boldsymbol{\lambda} \leq \boldsymbol{\theta}$ and $\boldsymbol{\theta}^+(0) = \boldsymbol{\mu}\,\mathbf{P} + \boldsymbol{\lambda} \geq \boldsymbol{\theta}$. It turns out the gap between $\boldsymbol{\theta}^-(t)$ and $\boldsymbol{\theta}^+(t)$ decreases geometrically with t:

Theorem 1. *The sequence* $\{\boldsymbol{\theta}^-(t) \mid t \geq 0\}$ *increases monotonically to* $\boldsymbol{\theta}$ *and the sequence* $\{\boldsymbol{\theta}^+(t) \mid t \geq 0\}$ *decreases monotonically to* θ, *with* $0 \leq \boldsymbol{\theta}^+(t) - \boldsymbol{\theta}^-(t) \leq \boldsymbol{\mu}\mathbf{P}(\mathbf{L}(\mathbf{I}-\mathbf{U})^{-1})^t$.

Theorem 1 follows directly from the following monotonicity property of the Gauss-Seidel operator:

Lemma 2. *If* $\tilde{\mathbf{x}} \leq \mathbf{x}$ *in* R_+^N $(\tilde{\mathbf{x}}, \mathbf{x} \geq 0)$, *then* $0 \leq G(\mathbf{x}) - G(\tilde{\mathbf{x}}) \leq (\mathbf{x} - \tilde{\mathbf{x}})\,\mathbf{L}\,(\mathbf{I}-\mathbf{U})^{-1}$.

The rate of convergence of the algorithm depends on the ordering of the state space, that is, the assignment of the indices 1, 2, ..., N to the N nodes. In the full paper we show that, for example, if the network is simply N queues in series, then there is an ordering under which the algorithm converges in one iteration and another under which the algorithm requires N iterations to converge. On this example, the Jacobi method and Goodman and Massey's method [M,GM] also require N passes to converge.

Intuitively, to speed the convergence of the Gauss-Seidel method we want to permute the rows and columns of P consistently so that most of the mass is above the diagonal. A heuristic, recently proposed by Greenberg and Vanderbei [GV] for finding such a state space ordering, works as follows. For each state i, the difference in the mass flowing out of that state and the mass flowing into the state is computed, $\sum_j p_{ij} - p_{ji}$, and a state for which this difference is maximal is selected and assigned the first position in the ordering. The row and column indexed by the maximal state are removed, and the procedure is repeated to select a state for the second position in the ordering, whereupon the row and column indexed by that state are removed, and the procedure repeated to select a third state, and so on.

REFERENCES

[GM] Goodman, J. and Massey, W., "The non-ergodic Jackson Network", *Journal of Applied Probability*, 21, 4, pp. 860,869 (December, 1984).

[GV] Greenberg, A. and Vanderbei, R., "On successive approximation methods for stochastic problems: probabilistic interpretation, ordering heuristics, and parallel processing," this proceedings (1988).

[J] Jackson, J., "Networks of waiting lines", *Operat. Res.*, 5, pp. 518-521 (1957).

[M] Massey, W., "Non-stationary queues", *Ph.D. Dissertation*, Department of Mathematics, Stanford University (1981).

[T] Tijms, H.C., *Stochastic Modelling and Analysis, a Computational Approach* 1986, John Wiley and Sons, New York.

[V] Varga, R., *Matrix Iterative Analysis*, 1962, Prentice Hall, Englewood Cliffs, NJ.

[W] Whitt, W., "Open and closed models of networks of queues", *AT&T Bell Laboratories Technical Journal*, 63, 9, pp. 1911-1980 (November, 1984).

A Load-Controlled Token Ring Protocol

Y. W. Chiang and C. C. Lee
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

ABSTRACT

Setting a constraint on station's token possession time such that both large file transfers and interactive processes are satisfied is a difficult task. In fact, no fixed token possession limit can satisfy all the traffic situations. This paper presents the idea of dynamically adjusting stations' token possession time based on the network traffic load. The goal of dynamic token time control is to enhance service efficiency of heavy users while maintaining the service rate of others. A simple dynamic token control mechanism that possesses this advantageous feature is presented and analyzed. We show analytically in two case studies and by simulation in a third case study that token ring with the presented load-driven access has substantially better performance than one without.

I. INTRODUCTION

Token ring has become an increasingly attractive protocol for LAN, for HSLN, and for internetworking applications. More importantly, thanks to IEEE 802 committee[1], IBM, and FDDI standards, token ring technologies are maturing rapidly. However, when a wide variety of communication demands have to be served, a problem shared by virtually all the networks that employ round-robin type of access strategies should be addressed. Specifically, to prevent one station from monopolizing the channel, a limit on station's token holding time should be incorporated in the access protocol. Indeed, IEEE 802.5[1] includes such considerations. The question is: how should this limit be chosen? If this limit is too high, interactive users or short messages would suffer since bulk file transfers or image transmissions could potentially dominate the network. On the other hand, if the time limit is too low, then not only it implies large token latency overhead but a user having large bandwidth demand would suffer even in the absence of other users. Since network traffic characteristics fluctuate with time, practically, there exists no elegant choice of a fixed time limit. Therefore, it is certainly desirable if this limit for station's token holding time can be dynamically adjusted in accordance with the latest network traffic demands. This dynamic feature would require two capabilities:

i. means of acquiring the latest load statistics
ii. rule of dynamically changing the time limit based on the latest load statistics.

Taking advantage of the unique feature of token ring that the ring interface of every station reads every bit circulating around the ring, we present in this paper a token control mechanism that offers these capabilities. The mechanism requires that a registration field be included in each transmission such that waiting stations can convey their demands. The objective of dynamic token time control is to offer a higher throughput to heavy users while maintaining the response time of others. In particular, the presented dynamic token control method was so designed that the token ring operation is not different than the conventional static token control method that employs a fixed token holding time when there are many busy stations in the ring. On the other hand, however, the dynamic element of the proposed method enables a heavy station (whose heavy status may be either persistent or instantaneous) to fully utilize the ring capacity for as long as other stations lack communication demands.

In performance evaluation, the presented token ring access protocol involves a difficult cyclic queue problem due to its dynamic nature. To illustrate the advantage of dynamic token time control, we have conducted three case studies. The first exploits the benefit received by a "single" busy station which has a large communication demand. We studied the performance using packets and messages as traffic arrival units, respectively. Analytical results are reported for this first case. The second case features an extremely heavy station and M regular stations which have Poisson packet arrivals. The throughput of the heavy station and the mean packet waiting time of the regular stations are performance characteristics of interest in this case. Analytical results were derived and are presented for this case also. In the third case, all the stations are assumed to have an identical message arrival rate but one of them has a very large mean message length while others have equal and short mean message length. The average message waiting time for the two categories of stations were considered as performance indices. These average waiting times versus channel utilization were obtained for both the conventional token ring and the presented load-driven token ring, respectively, by simulation. All three case studies show significant performance improvement achieved by dynamic token time control, especially when the token latency is critical. The paper concludes with potential extensions to more sophisticated load-driven token ring protocols.

II. A Dynamic Token Time Control Mechanism

To prevent heavy stations from dominating the token ring while other demanding stations are present, it is logical to impose a maximum token possession time on the ring access algorithm. In fact, this has been part of the IEEE 802.5 token ring standard. However, the choice of a proper time limit is a dilemmatic issue. Clearly, a larger token time limit benefits heavy users at the expense of short-message and interactive users while a smaller limit implies a large token latency overhead and thereby penalizes long message users and bulk file transfers. Since station traffic characteristics fluctuate and network application environment evolves, there does not exist a fixed time limit which is "proper" at all times. The idea of dynamic token time control presented in the following is to alleviate this dilemma.

Throughout this paper, we assume that the token ring operates with a single priority and uses the "multiple-token" strategy[2] for token transfer. Note that the network response time of an interactive user is closely related to the cycle time, i.e., the time between successive token possessions of a given station. Therefore, we shall impose a "basic time limit" on a station's transmission per token possession such that the worst-case cycle time is tolerable for interactive response. Nevertheless, as will be described below, there will be situations where this basic time limit does not apply to a transmitting station. These situations are where the dynamic control element contributes to the performance improvement.

The idea of using network traffic load to control token holding time can be realized in numerous ways, ranging from very simple to very complex. In the simplest case, the control strategy is basically the following: the transmitting station has to release the token at the end of the basic time limit if and only if any communication demand from other stations is detected by this station. If the transmitting station is lucky enough to exceed the basic time limit, then it has to release the token at the end of its current packet transmission once it detects another demand. The advantage of this simple strategy is limited to the situation where one heavy station is transmitting while all the others are momentarily idle. If there coexist two or more busy stations, this simple dynamic strategy performs no better than the static one whose token time limit equals our basic time limit. A more sophisticated dynamic token control strategy would be to determine the token holding time based on the number of busy users that coexist. This entails a more complex station registration mechanism (and substantially more overhead) and involves more complex design considerations. For simplicity in terms of design and analysis, we focus in this paper on the simple strategy mentioned above. For convenience, we shall call this simple dynamic token control mechanism the "dynamic ring access" (DRA) and the conventional token control method the "static ring access" (SRA) throughout this paper.

To realize the DRA, it simply requires that a "registration bit" be included in each data frame such that waiting stations can indicate their need by setting that bit while reading the data frame. In the context of IEEE token ring standard, the Access Control (AC) field[1] can be utilized. We note that the three priority bits and the three reservation bits in this field are unused when the ring operates with a single priority. One of the three reservation bits can be used as the registration bit. A station ready to transmit will set this registration bit when it passes by unless it has already been set by another station upstream. Upon detection of a set registration bit, the transmitting station must be prepared to release the token either at the end of the current packet transmission or as soon as the basic time limit is reached depending on whether the latter has been reached or not. There are certainly alternatives to using the AC field for registration and to addressing the problem within IEEE token ring standard.

The above discussion implies that the DRA is very easy to implement. With the DRA, a heavy station will be able to continue transmitting for as long as no other stations demand the channel. The benefit also applies to any station that instantaneously becomes the only busy station and requires more than the basic time limit. On the other hand, the presence of a heavy station in the ring causes no additional delay for the other stations for the DRA, as compared to the SRA. This is so because the rules governing token release for the DRA and the SRA are virtually the same when there exist multiple busy stations. Consequently, a performance edge can be expected for the DRA when an unbalanced station load is present. In the next section, we present some performance results that indeed justify the expectation.

III. PERFORMANCE ANALYSIS - THREE CASE STUDIES

In this section, we study the performance of the DRA and compare the results with the performance of the SRA. In terms of queueing analysis, we face basically a cyclic queue problem[3]. We note that, under certain network traffic conditions, the DRA and the SRA operate in an identical manner and thereby have the same performance. For example, if no station in the ring ever requires more than the basic time limit per token possession for its data transmission, then no performance edge can the token ring attain from the DRA. Therefore, we shall present three case studies aiming at exploiting the performance benefit that the DRA can offer over the SRA. We assume that packets of a fixed length X bits (including data and control bits) are transmitting units and that the token holding time is measured in numbers of packets. Let R *bits/sec* be the channel rate and thus P=X/R *sec* the packet transmission time. Let B *bit(s)* be the station latency and τ *sec* the loop propagation delay of the ring and thus $\tau'=(MB/R+\tau)$ *sec* the ring latency per cycle time. The ratio $a = \tau'/P$ is known as a critical parameter in network performance.

Case 1: Single busy station having heavy traffic

As was illustrated in Section II, the main goal of the DRA is to maximize the efficiency of serving a heavy enough station whenever all other stations are idle. To see the extent to which the DRA can accomplish this, we first consider the extreme case where there is a single busy station in the ring. We note that this case not only could very likely happen during "afterhours" but might occur very often, yet momentarily, during prime time. This case certainly leads to the maximum performance advantage that the DRA can achieve over the SRA.

Let the single busy station be Station 0 and M the number of other stations which are all idle. First suppose Station 0 has a Poisson packet arrival rate λ *packets/sec*. A basic time limit of 1 (packet) is assumed for both the DRA and the SRA. For the SRA, every packet transmission is accompanied by one ring latency time τ'. Hence if τ' is absorbed by the packet service time (i.e., service time=$P+\tau'$), then the ring behaves as an M/D/1 queue with constant vacation, the vacation time being τ'. As a result, the mean packet waiting time is given by[4]

$$w_S = \rho'(P+\tau')/2(1-\rho') + \tau'/2 = \rho(1+a)(P+\tau')/2[1-\rho(1+a)] \tag{1}$$

where $\rho'=\lambda(P+\tau')$ and $\rho=\lambda P$. For the DRA, the ring also behaves as an M/D/1 queue with constant vacation but the packet service time does not involve the ring latency. Hence, the mean waiting time is given by

$$w_D = \rho P/2(1-\rho) + \tau'/2 \tag{2}$$

Normalizing W_s and W_D by P yields

$$W_S = \rho(1+a)^2/2[1-\rho(1+a)] + a/2 \tag{3}$$

and

$$W_D = \rho/2(1-\rho) + a/2 \tag{4}$$

respectively. Observe that the maximum throughput is unity for the DRA and is 1/(1+a) for the SRA. The difference between W_S and W_D is dependent on a and ρ. Figure 1 shows the comparison of these two numbers for a=0.1. It is seen that the difference grows with ρ and explodes as ρ approaches 1/(1+a). This is expected since ρ represents the traffic load of Station 0 which has to be sufficiently heavy in order to capture the advantage of dynamic token time control.

Instead of Poisson packet arrivals and mean packet waiting time, we may also assume Poisson message arrivals and study the mean message waiting time even though transmissions are still on a packet by packet basis. The message length is assumed to be exponential with mean $1/\xi$ (bits) and is decomposed to packets of fixed length X. Therefore, instead of M/D/1, we have M/M/1 problems. For the SRA, the fact that each packet transmission is accompanied by one ring latency time τ' still holds. Consequently, the (normalized) mean message waiting time is given by

$$t_S = \{\rho(1+a)^2(1+e^{-\xi X})/2[1-\rho(1+a)]\} + a/2 \tag{5}$$

For the DRA, each message is transmitted in its entirety and thus does not invilve τ' in its service time. As a result, the mean message waiting time is given by

$$t_D = \{\rho(1+e^{-\xi X})/2(1-\rho)\} + a/2 \tag{6}$$

That the factor $(1+e^{-\xi X})$ appears in (5) and (6) implies that the advantage of the DRA over the SRA is even more significant than the packet arrival case. Indeed, even with a small ρ, packets arrive in bunches for the message arrival case, and the DRA enables Station 0 to empty its queue without spending any ring latency. As is shown in Figure 2, the differences between (5) and (6) are larger than those between (3) and (4).

Although this case study reflects some practical network traffic situation, it exploits simply the positive side of the DRA since only a single heavy station is in the picture. To examine the impact of the DRA on other stations, we need to bring the traffic to these stations.

Case 2: One Unlimited Station and M Poisson Arrival Stations

This case study involves two categories of stations: one extremely busy station (say, Station 0) and M "normal" stations each having a Poisson packet arrival rate λ. This case occurs when a station needs bulk file transfer and therefore has a tremendous channel demand. A gateway node is particularly likely to produce such a scenario. The case study is intended to show how the DRA can benefit Station 0 and how it impacts the normal stations. In terms of performance, it is certainly desirable to maximize the throughput of Station 0 while maintaining the response time of the others.

It is assumed that Station 0 has an unlimited number of packets to transmit. Hence, packet waiting time is not a performance index of interest to Station 0, the throughput is. In the following, a basic time limit of one (i.e., one packet) is assumed for the DRA; for the SRA, a variable time limit *k* will be considered. We note that for the SRA a larger *k* benefits Station 0 (throughput) at the expense of the others (packet waiting time).

First, consider the SRA with a limit-k service. Since Station 0 transmits k packets every cycle time, its channel consumption can be treated as ring overhead and be absorbed in the ring latency in calculating the mean cycle time and the mean packet waiting time for the normal stations. Let $\rho = M\lambda P$. To avoid infinite delay for the normal stations, it requires that

$$\rho < 1/[1 + (1 + a/k)/M] = \rho_{max} \qquad (7)$$

If (7) holds, then ρ is the throughput of the M normal stations and it is readily shown [3] that the average cycle time is $T_c = (kP+\tau')/(1-\rho)$. The throughput of Station 0 is thus given by

$$S_S(k) = kP/T_c = (1-\rho)/(1 + a/k) \qquad (8)$$

Using the approximation results derived by Fuhrmann[5] for a symmetric cyclic queue with limit-k service, the (normalized) mean packet waiting time is given by

$$W_S(k) \cong [(1-\rho)/(1 - \rho/\rho_{max})]\{[(k+a)(1 - \rho/M) + \rho/M]/2(1-\rho)\} \qquad (9)$$

where the approximation is perfect for k=1 and k=∞.

Now consider the DRA with a basic time limit of 1 (packet) for token possession. From the viewpoint of a waiting packet in a normal station, the ring operates just like the SRA with a limit-1 service, saving for the situation where this packet's arrival just missed the registration bit which is travelling back to the token holding station unset. Therefore, the mean packet waiting time for the normal stations is given by (9) with k=1:

$$W_D = [(1-\rho)/(1 - \rho/\rho_{max})]\{[(1+a)(1 - \rho/M) + \rho/M]/2(1-\rho)\} \tag{10}$$

where $\rho_{max} = 1/[1 + (1 + a)/M]$. The throughput for Station 0 is the complement of channel utilization attributed to the normal stations (i.e., ρ if $\rho < \rho_{max}$) including the ring latency and can thus be expressed as follows:

$$S_D = 1 - M\lambda(P + \tau'/N) = 1 - \rho(1 + a/N) \tag{11}$$

where N is the average number of packets transmitted by the normal stations per cycle time and is given by

$$N = N(M, \rho) = 1 + (M-1)\rho/\rho_{max} \tag{12}$$

We note that the first term represents the packet that initiates the cycle and the second term is the average number of packets generated by the normal stations in a cycle excluding the initiating packet. Also note that $N(1, \rho)=1$ for any ρ, that $\lim_{\rho\to 0}N(M, \rho) = 1$ for all M , and that $N(M, \rho_{max})=M$.

A graphical comparison between S_D and $S_S(k)$ is given in Figure 3 for k=1,3,10, with a=0.1. Analytically, comparing (11) and (8) reveals that

$$S_D > S_S(k) \quad \text{unless} \quad k > N(1 - \rho - \rho a/N)/\rho \equiv k_o \tag{13}$$

where N is given by (12). This implies that the SRA needs to use at least a limit-k_o service in order to achieve the same throughput that the DRA offers to Station 0. Since $W_D = W_S(1)$ and $W_S(k) \cong kW_S(1)$ (if M is large), we conclude that the mean packet waiting time for the normal stations in the SRA is about k_o times as large as that in the DRA while both the SRA and the DRA maintain the same throughput for Station 0. The relationship between k_o and ρ is displayed in Table 1 for a=0.1. It is seen that k_o explodes at low ρ.

Case 3: Same Message Arrival Rate But One has Very Long Messages

Considering message arrivals and mean message waiting time may be more meaningful in practice though involve less tractable analysis. In this section we consider a case where all the stations in the ring have an identical message arrival rate λ but one of the stations has a very large mean message length X_1 while all the others have an identical and short mean message length X_0. A fixed packet size X is used. Messages which are shorter than X still takes one full packet each for transmission and messages longer than X are each decomposed into multiple packets for transmission. We are interested in the average message waiting time for each class of stations. Specifically, we show by simulation results how the DRA can benefit the token ring performance under the assumed traffic condition. The following parameter values were used in the simulations: M = 100 stations; R = 10 Mb/sec; X = 1,024 bits; B = 1 bit; $\tau = 10^{-5}$ sec; X_0 = 1,000 bits; X_1 = 500,000 bits. One of the 100 stations is assumed to generate long messages while all the others generate short messages only. We have made an attempt to maintain the mean message waiting time for short message stations at a comparable level between the DRA and the SRA so that performance advantage of the DRA can directly reveal in the mean message waiting time for the long message station. Therefore, a limit-1 service is assumed for the SRA. Table 2a and Table 2b show the simulation results: D_1 denotes the average message delay for the station having long messages, D_0 the same for the other stations, and D the same for all the stations.

The values of ρ, D_0, D_1, and D were all obtained from simulations. In the last case for the SRA (bottom raw of Table 2b) where $\rho=0.873$, the token ring is actually saturated. In this case, the average message delay for the long messages D_1 is actually infinity (so is D). However, in calculating D_1 and D, only those messages that have been successfully transmitted at the end of simulation were included. The backlogged messages at the end of simulation were not accounted for. That was why finite D_1 and D were obtained. It is observed that the DRA not only greatly benefits the long-message station but slightly benefits the others. In addition, by saving a significant amount of latency overhead, the DRA also offers more useful channel capacity (i.e., maximum throughput) then the SRA.

IV. CONCLUSION

We have presented and analyzed in three case studies a simple dynamic token control method and showed that it can greatly benefit a single heavy station without sacrificing others. To extend the advantage to multiple busy stations, more sophisticated load-driven ring access methods have to be developed. As long as the maximum waiting time for a packet arriving at an empty station queue is maintained at an acceptable level, the interactive and short message users would not be appreciably penalized. Since a token ring usually uses a monitoring station to perform some network maintenance and management tasks such as fault detection, token recovery, data removal, etc., this monitoring station can also be used as control center to collect traffic statistics and use them to dynamically determine (and broadcast to all the stations) the token holding time. This approach should facilitate the design of more comprehensive dynamic token control mechanisms.

REFERENCES

1. IEEE 802 Committee, Token Ring Access Method and Physical Layer Specifications, IEEE Std. 802.5-1985.
2. W. Bux, "Local Area Subnetworks: a Performance Comparison," IEEE Trans. Communications, vol. COM-29, Oct. 1981, pp. 1465-1473.
3. H. Takagi, Analysis of Polling Systems, MIT Press, Cambridge, MA, 1986.
4. D. Bertsekas and R. Gallager, Data Networks, Prentice-Hall, 1987.
5. S. W. Fuhrmann, "Symmetric Queues Served in Cyclic Order," Operations Research Letters, vol. 4, Oct. 1985, pp. 139-144.

Figure 1. Performance Comparison of SRA and DRA: Case1-Packet

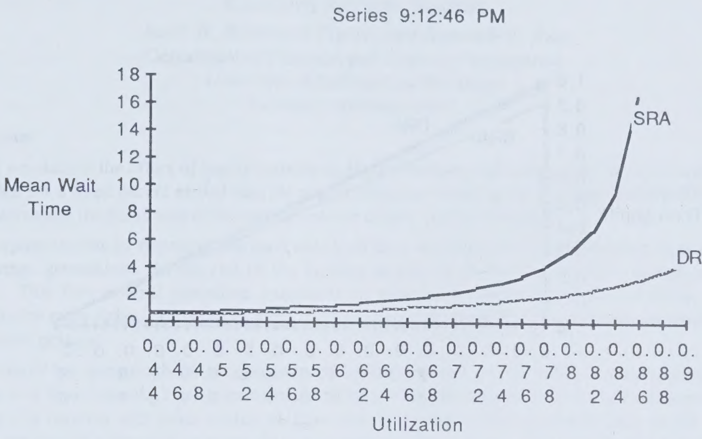


Figure 2. Performance Comparison of SRA and DRA: Case1-Message

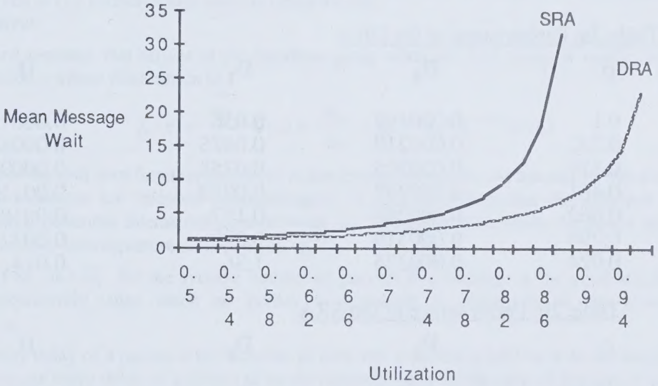


Table 1: Limit-ko SRA has the same throughput as the DRA

ρ:	0.001	0.01	0.02	0.08	0.2	0.5	0.8	0.9	0.98	ρ_{max}
k_o:	1100	200	147	103	84	51	20	10	2	1

Figure 3. Performance Comparison of SRA and DRA: Case 2

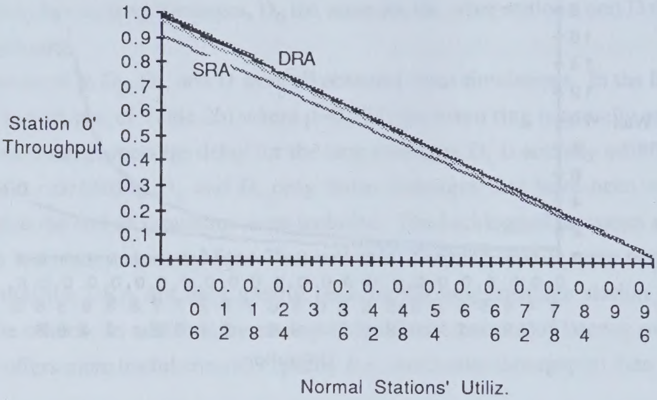


Table 2. Simulation Results for Case 3

Table 2a: Performance of the DRA

λ	ρ	D_0	D_1	D
1.5	0.1	0.000192	0.056	0.000754
3.5	0.232	0.000210	0.0675	0.000882
5.0	0.329	0.000225	0.0758	0.000976
7.0	0.461	0.000248	0.0952	0.00119
10	0.662	0.000288	0.157	0.00185
12	0.794	0.000319	0.292	0.00325
14	0.924	0.000335	1.36	0.014

Table 2b: Performance of the SRA

λ	ρ	D_0	D_1	D
1.5	0.0972	0.000207	0.066	0.000855
3.5	0.234	0.000227	0.083	0.00106
5.0	0.322	0.000241	0.0914	0.00114
7.0	0.462	0.000265	0.129	0.00153
10	0.657	0.000310	0.246	0.00276
12	0.79	0.00034	0.653	0.00685
14	0.873	0.000368	104	0.971

ACKNOWLEDGEMENT

This work was supported in part by the Institute for Modern Communications (IMC) at Northwestern University. We would like to thank Mr. E. Drakopoulos for his help with preparing some numerical results.

Use of Flow Control in a Finite User Random Access System

Amir B. Behroozi-Toosi, and Ramesh R. Rao
Department of Electrical and Computer Engineering
University of California at San Diego
La Jolla, California 92093

1. Introduction

In [1] we studied the effect of bursty arrivals on the performance of a finite user multiple access system. We found that for a fixed packet arrival rate the packet delay decreases as the burstiness of the arrival process increases. Increasing the burstiness of the arrival process is thus clearly desirable.

We implement this by requiring that each user hold back messages for a fixed duration in anticipation of further message generations. At the end of the holding period all of the accumulated message arrivals are ''released''. This flow control procedure introduces an additional component of packet delay, henceforth referred to as the entry delay. The holding time is chosen so as to ensure that the transformed message arrival process remains poisson.

In section 4 we compare the total delay induced by the proposed flow control scheme with the total delay in the absence of flow control. The comparison identifies the region in which flow control reduces the average packet delay. In contrast with other studies on flow control [2] that minimize transit delay at the expense of entry delay we consider the total queuing delay experienced by a packet. Thus our results will facilitate an exact performance comparison between the proposed scheme and other schemes like TDMA.

2. Model of the Communication System

The model of the communication system is essentially the same as in [1] except that now each user has two queues, an initial interface queue followed by the service queue. The action of the interface queue, explained in the next section, makes the arrivals to the channel more bursty. The service queues employ the algorithm given in [1]; further details may be found therein.

3. Flow Control

The first message that arrives at the interface queue when the said queue is empty, is held back for a random duration x whose distribution is

$$f_x(x) = \frac{\alpha_o}{\alpha_i}\,\delta(x) + \frac{\alpha_o}{\alpha_i}\,(\alpha_i - \alpha_o)\,e^{-\alpha_o x}U(x)$$

where $U(x)$ is the unit step function and $\delta(x)$ is the impulse function. At the end of this duration all of the accumulated messages are released simultaneously. It can be shown that the interface queue receives messages with exponential interarrival time of mean α_i^{-1} and then it transmits the newly grouped messages with an exponential interdeparture time of mean α_o^{-1}.

Now if $\overline{m}_i$ and $\overline{m}_o$ are the average number of packets in a message at the input and the output of the processor respectively, then since no packet is generated or destroyed in the processor we have $\overline{m}_i\ \alpha_i = \overline{m}_o\ \alpha_o$.

The entry delay of a packet is the duration of time that a packet is held back in the interface queue. We define the average entry delay of a packet to be the expected value of the ratio of the sum of the waiting times of all packets in the interface queue that arrived during x to the the total number of packets that arrived. The average delay of a packet $E[d]$ satisfies the following relation:

$$E[d] \equiv E\left[\frac{\sum_{k=0}^{N(x)} m_k\ d_k}{\sum_{k=0}^{N(x)} m_k}\right] = \frac{1}{2}\left(\frac{\alpha_i - \alpha_o}{\alpha_i\ \alpha_o} + \frac{\alpha_i - \alpha_o}{\alpha_i\ (\alpha_i + \alpha_o)}\right)$$

In the case of a single user in a homogeneous system with M users we have the following simple expression for the average delay incurred by a packet in the interface queue.

$$E[d] = \frac{M}{2\lambda}(\overline{m}_o - \overline{m}_i)\left(1 + \frac{\overline{m}_i}{\overline{m}_i + \overline{m}_o}\right) \qquad \text{with,} \quad \overline{m}_i = \frac{\lambda}{M\ \alpha_i}, \text{ and } \ \overline{m}_o = \frac{\lambda}{M\ \alpha_o}$$

This work was sponsored by the SDIO/IST and managed by the Office of Naval Research under contract ONR N00014-85-K-0508.

4. Total Delay

Noting that the channel arrival process remains compound Poisson with rate α_o and average message length of $\overline{m}_o$, the total delay can be computed based on the results of [1].

As shown in in Fig. 1, the region in which contentions first manifest themselves is also the region in which the flow control mechanism is most effective. This is where the bunching of messages pays off. As is to be expected the flow control processor does not improve the delay performance at low throughputs. This is because at low throughput the channel rarely sees a collision and the additional waiting time imposed on messages in the interface queue does not serve any purpose. For a known $\overline{m}_i$ one can construct a set of delay curves for different values of $\overline{m}_o$. Then for a fixed arrival process (fixed λ and $\overline{m}_i$) one can search for an optimum value for $\overline{m}_o$ to minimize the total average delay of a packet. Finally, Fig. 2 shows the envelope of the achievable average delay if the controller knows the arrival process parameters and chooses an appropriate $\overline{m}_o$. At low throughputs the controller essentially does not alter the arrival process and at higher throughputs it selects different $\overline{m}_o$ depending on the value of λ to minimize the total average delay.

5. Conclusions

In this paper, we have examined the role of flow control on the performance of the blocked access stack algorithm modified for a finite number of users. It was found that bunching the messages at the entrance to the network can reduce the total delay for certain values of throughput. The total delay, which includes the bunching delay, is reduced mainly when the bunching action reduces the probability of a collision. Our calculations facilitate comparison between the proposed scheme and other schemes like TDMA (assuming geometrically distributed message lengths [3]). One such comparison is shown in fig.2. In addition, our results show that characteristics of the arrival process and the number of users are important factors in establishing the optimality of TDMA or random access algorithms.

References

[1] Amir Behroozi-Toosi and Ramesh Rao, "Effect of bursty arrivals on the performance of Multiple Access System", *Proceedings of the 1988 CISS,* Princeton, New Jersey, March 1988.

[2] Mario Gerla and Leonard Kleinrock, "Flow Control: A Comparative Survey", *IEEE Trans. Commun.*, vol. COM-28, pp. 553-574, April 1980.

[3] Jeremiah F. Hayes, *Modeling and Analysis of Computer Communication Networks,* Plenum Press, New York, 1984.

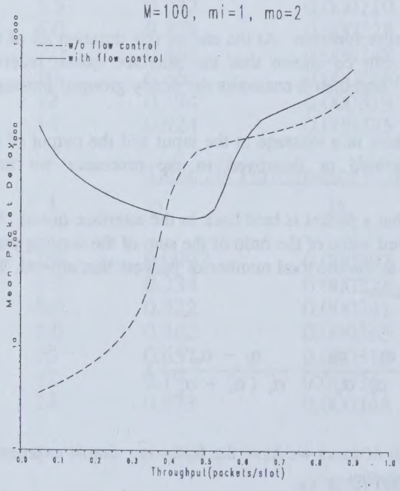


Figure 1.

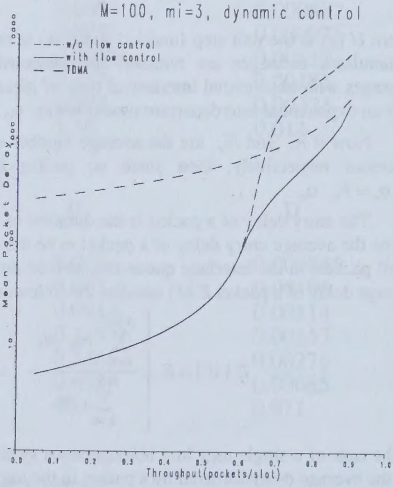


Figure 2.

PARALLEL TIME COMPLEXITY OF ROUTING IN PERMUTATION NETWORKS

David M. Koppelman
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute

A. Yavuz Oruç
Electrical Engineering Department
University of Maryland

Abstract

This paper examines the programming complexity of permutation networks by using a simple dependency model of computation and certain recurrences that describe the relations between the programming times of various parts of a permutation network. Lower programming time complexity bounds are derived by revealing certain dependencies between permutations with which a given network is programmed and the permutations that are assigned to the parts of that network. These bounds are extented to recursively decomposed Clos networks directly by using recurrences that relate programming time of a Clos network to the programming times of its parts, and it is shown that, for certain recurrences, a higher degree of parallelism at each level of recursion does not mean a lower programming time complexity.

1. Introduction

This paper focuses on programming permutation networks. (The reader interested in permutation networks and their applications can refer to [4, 6].) Permutation networks are often built recursively from smaller permutation networks so that the programming problem takes the following form:

Given a permutation P that defines a one-to-one mapping of a network's inputs to its outputs, obtain a set of permutations $p_1, p_2, \ldots, p_k$ such that the ith subnetwork realizes permutation p_i; $1 \leq i \leq k$ in order for the original network to realize P.

An efficient solution to this problem is very critical to the performance of the network in question since the programming time, along with the network's path-length, determines how fast paths can be established through the network. As a result, this problem has received a considerable interest in the literature, and both serial [9, 11] and fast parallel set-up algorithms [5, 8] have been reported for permutation networks. These algorithms run on a variety of machines. For example, the looping algorithm developed in [9, 11] runs on a serial machine. On the other hand, the parallel programming algorithms use various parallel computer organizations such as shared memory, mesh–connected, perfect shuffle, and cube–connected architectures [2,7,10].

In this paper, we shall be concerned with the complexity of programming permutation networks. Apart from the known time bounds for parallel programming algorithms for permutation networks on various parallel computers cited above, not much is known about the complexity of this problem. These parallel algorithms at best lead to upper bounds for the time and hardware complexities of the programming problem. Furthermore, the parallel computer models used in some of these efforts are not suitable for hardware implementation, and we need to use a more hardware oriented model for our complexity analysis.

The model we use is a simple directed acyclic graph (DAG) with restricted fan-in, that is the number of edges directed towards a vertex is bounded. The nodes of this graph are assigned combinational functions which we need not specify for the purposes of deriving our bounds. The directed acyclic graph has distinct vertices through which the specified permutation enters the computation. It also has distinct vertices from which the computed permutations for the subnetworks of the original network are collected. Thus, for a permutation of size n, the graph will have n input vertices whereas the number of output vertices is arbitrary, and does not affect our complexity bounds.

In deriving the complexity bounds, we shall consider two cases: In the first case, the network to be programmed is specified as a collection of some subnetworks which are not decomposable and

---

This work is supported in part by the National Science Foundation under grant No:CCR-8708864

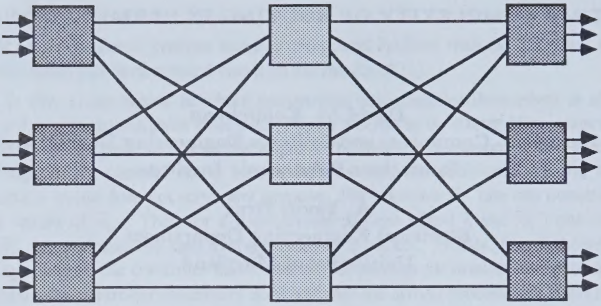


Figure 1. A Clos network with $r = m = 3$.

interconnected in some prespecified fashion. The task of a programming graph is then to compute the maps for these subnetworks. Our model of permutation network in this case is a 3-stage Clos network [3]. In the second case, we shall allow the original network to be specified recursively so that each of its parts can be decomposed as many times as possible, i.e., until it reduce to a trivial network. In this case, we shall examine the limits of speed-up one can hope to have in programming recursively constructed permutation networks. This shall be carried out by dividing the programming algorithms into classes based on the recursive structures of the networks to be programmed. It will be shown that the programming algorithms within a class will all exhibit the same degree of large grained parallelism and the same computational complexity within grains. Algorithms within a class also exhibit similar overall time complexity and hardware (space) complexity. In addition, we shall point out some interesting anomalies regarding the impact of small and large grained parallelism on the overall time complexity of the programming algorithms.

The remainder of the paper is organized as follows: In Section 2, we introduce the notation used in the paper, and state a few preliminary facts. In Section 3, we introduce the notion of dependency between the inputs and outputs of a programming graph. To find a bound on programming time as many dependencies as possible is found for an output; the programming time is proportional to the logarithm of that number. These dependencies are found by showing that for certain pairs of permutations the output must be different under each. With these bounds established, we then discuss the time and space bounds for several classes of programming algorithms in Section 4. Finally, the paper is concluded in Section 5.

2. Notation and Preliminary Facts

A Clos network consists of r input and r output stage switches of m inputs each, and m center stage switches of r inputs each [3]. The Clos network with $r = m = 3$ is shown in Figure 1. The inputs and outputs to the network's switches will be referred to as follows: $I(x, y)$ refers to the y'th input (or output) to the x'th input stage switch; if $I(x, y)$ is on the left side of an equal sign then inputs are being referred to, if $I(x, y)$ is on the right side of an equal sign then outputs are being referred to. Similarly, the inputs and outputs of the center and output stage switches are referred to as $M(x, y)$ and $O(x, y)$ (M and O stand for middle and output respectively). Thus writing $M(1, 2) = I(2, 3)$ implies that the third output of the second switch in the input stage is connected to the second input of the first switch in the middle stage.

Using this notation, the connections among a Clos network's switches are

$$M(i, j) = I(j, i) \quad \text{and} \quad O(j, i) = M(i, j)$$

for $0 \leq j < r$ and $0 \leq i < m$. It is possible to set one input or output stage switch to a particular state permanently without sacrificing rearrangeability [11]. In this paper, without loss of generality, it will be assumed that the top most switch in the input stage is permanently set to the identity state. Note, that setting this switch to identity does not change the notation, inputs and outputs from that switch are still referred to as $I(0, y)$.

For purposes of analysis, the programming hardware for a Clos network will be two DAGs with mr inputs and mr outputs, one for programming the input stage switches and one for programming the middle stage switches. The output stage is not treated separately since computing the map for that stage takes no longer than computing the map for the input stage switches. The mr inputs to a DAG describes the permutation to be realized by the entire Clos network and the mr outputs describe the permutation to be realized by the center or the input stage switches.

The outputs of the DAG for the input stage will be referred to as $P_I(x, y)$ for $0 \leq x < r$ and $0 \leq y < m$; $P_I(x, y)$ is set to the output of input stage switch x to which $I(x, y)$ is mapped to. $P_M(x, y)$ is defined similarly for the middle stage.

Both DAGs receive as input the permutation to be realized and the inputs to each DAG will be referred to in the same way as the permutation $P(x, y)$, $0 \leq x < r$, $0 \leq y < m$. $P(x, y)$ is the output in the output stage of the programmed network to which input $I(x, y)$ in the input stage is mapped to.

Note that, in general, for any one permutation there may be many different P_I's and P_M's that can result in a properly programmed network. A $P_I(x, y)$ or $P_M(x, y)$ which results in a properly programmed network will be called *valid*; otherwise it will be called *invalid*. A network which is programmed by using an invalid $P_I(x, y)$ or $P_M(x, y)$ would not realize the permutation desired.

It is possible to derive a lower bound on the amount of time it would take for a programming DAG to compute its outputs. This lower bound is obtained by examining the Clos network structure to find dependencies between inputs and outputs of the corresponding programming DAG. An output of a DAG is said to be *dependent* upon an input if there exists a connection between that input and that output and whose removal results in an invalid value for that output. Dependencies will be described in more detail below. Once these dependencies are found, a minimum computation time can be determined by using the following lemmas:

Lemma 1: Suppose an output, Δ, of the programming DAG for the input or middle stage, depends upon z of the DAG's inputs. Suppose also that every vertex in the DAG has a fan-in of no more than w. Then the minimum amount of time to compute Δ is $\log_w z$. ||

Lemma 2: An output, Δ, of the programming DAG for the input or middle stage is dependent upon a subset $\mathcal{P}$ of inputs if there exist two permutations which differ only on inputs in $\mathcal{P}$, and the set of preimages of Δ under P is disjoint from the set of preimages of Δ under P'. ||.

It will be shown in the next section that P and P' exist to establish dependencies for a Clos network. First, some general theorems about dependencies are proven.

Suppose it is shown that an output of a programming DAG is dependent upon a set of inputs $\mathcal{P}$. That output is obviously then dependent upon at least one of the members in $\mathcal{P}$. Suppose it is shown that an output is dependent upon two sets of inputs, $\mathcal{P}_1$ and $\mathcal{P}_2$ and that $\mathcal{P}_1 \cap \mathcal{P}_2 \neq \phi$. Then the output can still be dependent upon only one input (one of the inputs in the intersection). If $\mathcal{P}_1 \cap \mathcal{P}_2 = \phi$ then the output depends upon at least two inputs.

The following two theorems will generalize these examples of dependency.

Theorem 1: Given an output, Δ,of a programming DAG, an integer x, and a set of inputs, denoted $\mathcal{U}$, then if for all $\mathcal{P} \subset \mathcal{U}$ $|\mathcal{P}| = \alpha$ and output Δ is dependent upon $\mathcal{P}$, then output Δ is dependent upon at least $|\mathcal{U}| - \alpha + 1$ members of $\mathcal{U}$.

Proof: Suppose that Δ is dependent on less than $|\mathcal{U}| - \alpha + 1$ members of $\mathcal{U}$, or stated another way, suppose that there are at least α members of $\mathcal{U}$ that Δ does *not* depend upon. Let $\mathcal{Q}$ be the set of all members of $\mathcal{U}$ which Δ does not depend on. If $\mathcal{Q} = \alpha$ then by the hypothesis, Δ depends on $\mathcal{Q}$ which is a contradiction. On the other hand, if $|\mathcal{Q}| > \alpha$, then by the hypothesis, Δ depends on every α-subset of $\mathcal{Q}$ leading to a contradiction. ||

For example, suppose $\mathcal{U}$ contains 10 elements and $x = 5$. If an output Δ is dependent upon every subset of $\mathcal{U}$ containing 5 elements then there must be at least 6 elements in $\mathcal{U}$ which Δ is dependent upon.

A similar result which will also be needed is the following:

Theorem 2: Given an output, Δ, of a programming DAG,and z disjoint sets of inputs $\mathcal{U}_0 \ldots \mathcal{U}_{z-1}$ then if Δ depends upon $\mathcal{U}_i \cup \mathcal{U}_j$ for all $0 \leq i < j < z$ then Δ depends upon at least all but one of the $\mathcal{U}_i$. ||

3. Lower Bounds for the Programming Time of a Clos Network

In this section, the above theorems will be used to find the number of inputs which are directly connected to a particular output in a Clos network's programming hardware given that the dependencies between *sets* of inputs and an output is known.

Definition 1: Given a permutation P, the sets $H_{i,j}$ for a Clos network are defined in the following way:

$$H_{i,j} = \{x | P(i,x) = O(j,y) \ \ 0 \le y < m\} \tag{1}$$

In words, the set $H_{i,j}$ contains those inputs of switch i in the input stage which P maps to outputs of switch j in the output stage. ||

Definition 2: A permutation P is said to contain a *loop* $(a_0b_0a_1b_1\ldots a_{l-1}b_{l-1})$ of length l if there exist distinct integers a_i and distinct integers $b_i; 0 \le a_i, b_i \le r-1$ for $0 \le i < l$ and sets of inputs $F(a_i)$ and $G(a_i)$ for $0 \le i < l$ such that

$$F(a_i) = H_{a_i,b_i}, \quad G(a_i) = H_{a_i,b_{i-1 \bmod l}}. \tag{2}$$

||

Definition 3: A loop $(a_0b_0a_1b_1\ldots a_{l-1}b_{l-1})$ is said to be *balanced* if

$$|F(a_i)| = |G(a_i)| = \frac{m}{2} \tag{3}$$

for $0 \le i < l$. ||

Let $P(i, 0\ldots m-1)$ denote the restriction of permutation P to the inputs $I(i,j); 0 \le j \le m-1$. Then the following example illustrates a loop of length 3 where $m = 4$ and $r = 3$:

Example:

$$\begin{aligned}
P(0,0\ldots 3) &= (3,0)(3,2)(3,3)(3,1) \\
P(1,0\ldots 3) &= (0,1)(0,2)(2,1)(2,3) \\
P(2,0\ldots 3) &= (0,3)(1,2)(0,0)(1,3) \\
P(3,0\ldots 3) &= (2,0)(1,0)(2,2)(1,1).
\end{aligned}$$

This permutation contains one loop of length 3, that is (1,0,2,1,3,2). The values of a_i and b_i are specified as $a_0 = 1, a_1 = 2, a_2 = 3$, and $b_0 = 0, b_1 = 1, b_2 = 2$. The sets $F(a_i)$ and $G(a_i)$ for this loop are:

$F(1)=\{0,1\}$ $G(1)=\{2,3\}$
$F(2)=\{1,3\}$ $G(2)=\{0,2\}$
$F(3)=\{0,2\}$ $G(3)=\{1,3\}$

Note that sets $F(0)$ and $G(0)$ are not defined because they are not contained in a loop. ||

Definition 4: A *fixed balanced loop* of length l, which will simply be called a fixed loop from now on, is a loop in which $a_0 = 0$ and $0 \in G(0)$. ||

As stated earlier it is assumed that the top most switch, i.e., switch 0, in the input stage of a Clos network will always be set to the identity permutation permanently.

The permutation example given above does not contain a fixed loop.

Given a permutation P containing a loop with the usual parameter names, let $S(i)$ be the set of middle stage switches through which the signals which start at inputs $I(a_i, z)$ for $z \in F(a_i)$ are routed. Let $S^{-1}(i)$ be the set of center stage switches through which the signals which start at $I(a_{i+1}, z)$ are routed where $z \in G(a_{i+1})$.

Theorem 3: Given a Clos network with parameters m and r, where m is even, and a permutation P which contains a fixed loop then $S(i) = S(j)$ for $0 \le i \le j < l$ and there exists exactly one possible set $S(0)$.

Proof: Since $0 \in G(0)$ $F(0)$ can be chosen in only one way, and since input stage switch 0 is permanently set to identity, $S(0)$ is unique, that is, it can only be chosen one way. Because of the structure of a Clos network $S(i) \cap S^{-1}(i) = \phi$ if $0 \le i < l$. In addition, since all $S(i)$ and $S^{-1}(i)$ contain exactly $m/2$ elements by the definition of F and G, $S(i) \cup S^{-1}(i) = \{0, 1, \ldots, m-1\}$, which is the universe over which S and S^{-1} are defined. Therefore $S(0) = S(1) = \ldots = S(l-1)$. ||

A fixed loop of length 3 is illustrated in Figure 2. The permutation which contains that fixed loop is

$$(1,0)(0,0)(0,3)(1,2) \ (0,1)(2,2)(2,3)(0,2) \ (1,1)(1,3)(2,0)(2,1)$$

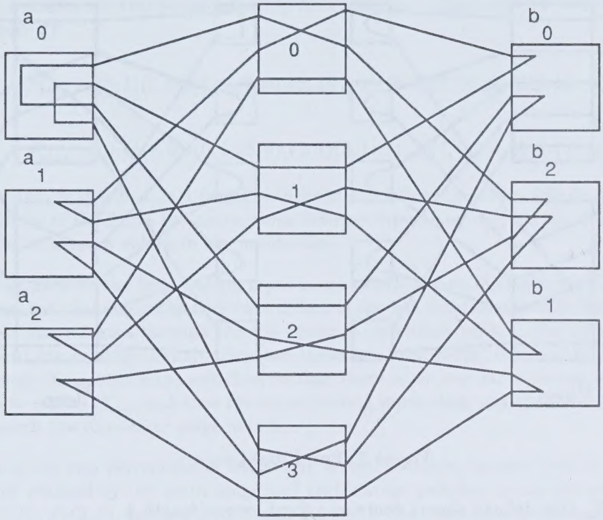


Figure 2. An illustration of Theorem 3 on a Clos network with $r = 3$and $m = 4$.

The values for a_i and b_i are: $a_0 = 0$, $a_1 = 1$, $a_2 = 2$, $b_0 = 0$, $b_1 = 2$, and $b_2 = 1$. The set $S(0) = \{1, 2\}$ and the set $S^{-1}(0) = \{0, 3\}$. The F and G sets are: $F(0) = \{1, 2\}$, $F(1) = \{1, 2\}$, $F(2) = \{0, 1\}$, $G(0) = \{0, 3\}$, $G(1) = \{0, 3\}$, and $G(2) = \{2, 3\}$.

Using this property of a fixed loop, dependencies between inputs and outputs of a programming DAG can be derived by finding two permutations, P_{loop} and P'_{loop}, both containing fixed loops of length 4, but one loop being a rearranged version of the other. More formally:

Definition 5: Given a permutation P_{loop}, of inputs to outputs of a Clos network with parameters m (m even) and $r > 3$ containing a fixed loop of length 4 with parameters $a_0 = 0, a_1, a_2, a_3$, b_i, $F(a_i)$, and $G(a_i)$ for $0 \leq i < 4$, permutation P'_{loop} is defined in the following way:

First, create a set T_1 containing all possible pairs, (x, y) where $x \in G(a_1)$, and $y \in F(a_3)$, such that for any two pairs, (x_1, y_1), (x_2, y_2), if $x_1 = x_2$ then $y_1 = y_2$ and if $x_1 \neq x_2$ then $y_1 \neq y_2$.

Then

$$P'_{\text{loop}}(a_1, x) = P_{\text{loop}}(a_3, y), \tag{4}$$

for all $(x, y) \in T_1$.

Next, create a set T_2 containing all possible pairs, (x, y) where $x \in F(a_3)$, and $y \in G(a_1)$, such that for any two pairs, (x_1, y_1), (x_2, y_2), if $x_1 = x_2$ then $y_1 = y_2$ and if $x_1 \neq x_2$ then $y_1 \neq y_2$.

Then

$$P'_{\text{loop}}(a_3, x) = P_{\text{loop}}(a_1, y), \tag{5}$$

for all $(x, y) \in T_2$. ||

Unspecified maps in P'_{loop} are equal to the corresponding maps in P_{loop}. For example, the following two permutations for $m = 2$, $r = 5$ constitute a pair of P_{loop} and P'_{loop} :

$$P_{\text{loop}} = (0, 1)(1, 0), (1, 1)(2, 0), (2, 1)(3, 1), (0, 0)(3, 0), (4, 0)(4, 1),$$

$$P'_{\text{loop}} = (0, 1)(1, 0), (0, 0)(2, 0), (2, 1)(3, 1), (1, 1)(3, 0), (4, 0)(4, 1),$$

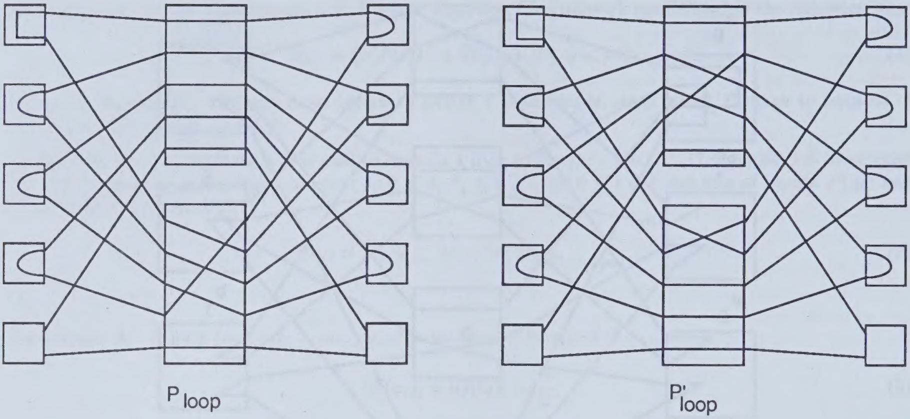


Figure 3. Two Fixed Loops.

Theorem 4: P'_{loop} (as defined above) contains a fixed loop of length 4.

Proof: To prove this theorem the complete set of sets $H_{i,j}$ (as defined in (1)) will be listed for this permutation, then it will be demonstrated that these sets satisfy the requirements set forth in the definition of a loop. Let $H'_{i,j}$ be the sets, as defined in (1), corresponding to P'_{loop}, and let $H_{i,j}$ be the sets corresponding to P_{loop}. Because only a portion of P'_{loop} is different then P_{loop} the following sets are equal: $H'_{i,j} = H_{i,j}$ for the following pairs of i and j :

$$(i,j) \in \{(0,b_3),(0,b_0),(a_1,b_1),(a_2,b_1),(a_2,b_2),(a_3,b_2)\}.$$

The remaining sets contain outputs to which inputs not already delt with, entering input stage switches a_1 and a_3, are mapped to. The inputs to input stage switch a_1, which in P_{loop} were mapped to b_0 are, in P'_{loop}, mapped to b_3. Similarly those inputs entering input stage switch a_3 which are mapped to outputs in b_3 in P_{loop} are mapped to outputs in b_0 in P'_{loop}. Thus it can be seen that $H'_{a_0,b_3} = H_{a_0,b_1}$ and it can be seen that $H'_{a_3,b_1} = H_{a_3,b_3}$. Let $a'_0 = 0$, $a'_1 = a_3$, $a'_2 = a_2$, and $a'_3 = a_1$, $b'_0 = b_0$, $b'_1 = b_2$, $b'_2 = b_1$, $b'_3 = b_3$. Substituting these values for a' in the labels of the H' sets and then assigning F and G labels yields:

Using a	Using a'	F and G assignments.
H_{a_0,b_3}	$H_{a'_0,b'_0}$	$G(0)$
H_{a_0,b_0}	$H_{a'_0,b'_0}$	$F(0)$
H_{a_1,b_1}	$H_{a'_3,b'_2}$	$G(a'_3)$
H_{a_2,b_1}	$H_{a'_2,b'_2}$	$F(a'_2)$
H_{a_2,b_2}	$H_{a'_2,b'_1}$	$G(a'_2)$
H_{a_3,b_2}	$H_{a'_1,b'_1}$	$F(a'_1)$
H_{a_3,b_0}	$H_{a'_1,b'_0}$	$G(a'_1)$
H_{a_1,b_3}	$H_{a'_3,b'_3}$	$F(a'_3)$

The first column in the table lists the sets $H_{a,b}$ in permutation P'_{loop} using a and b for the subscript and parameter. In the second column a' and b' values are used. In the last column the sets, $H_{a,b}$ are renamed to F and G consistent with the definition of a fixed loop.

The assignment of the label $G(0)$ is not arbitrary because only the set H_{0,b'_3} contains the input $I(0,0)$. Since it is possible to find the sets consistent with the definition, P'_{loop} contains a loop of length 4. ||

For example, consider the two permutations given earlier for a Clos network with $m = 2$, and $r = 5$ (illustrated in Figure 3):

$$P_{\text{loop}} = (0,1)(1,0),(1,1)(2,0),(2,1)(3,1),(0,0)(3,0),(4,0)(4,1),$$

$$P'_{\text{loop}} = (0,1)(1,0),(0,0)(2,0),(2,1)(3,1),(1,1)(3,0),(4,0)(4,1),$$

where the permutation is specified as a list of outputs in the following order: $P(0,0)$ $P(0,1)$, $P(1,0)\ldots$ $P(4,1)$. In both halves of the figure the connections between inputs and outputs are shown as loops. The loops start and end at the top switch in the input stage.

Notice that the parts of the loop leaving input stage switch 2 have the same destinations in the left and right diagrams, but that the paths they take differ: in the left diagram the signal leaving input stage switch 2 for output stage 2 goes through the top center stage switch while in the right diagram it goes through the bottom center stage switch. Because these are fixed loops, there is only one way for the signals to go through the center stage switches, so that there is no way of programming the network in the right diagram to realize P'_{loop} such that the signal leaving input stage switch 2 bound for output stage switch 2 goes through the top center stage switch.

Note also, that the two permutations differ only in two switches, namely $P(3,0)$ and $P(1,0)$. The permutations to be realized by the input stage and center stage switches in the left and right diagrams must differ in $P_I(2,0)$, $P_I(2,1)$, and in $P_M(0,2)$ and $P_M(1,2)$. Thus there is a dependency between $P(3,0)$ & $P(1,0)$ and $P_M(0,2)$ & $P_M(1,2)$. This technique of using P'_{loop} to find dependencies is generalized below:

Theorem 5: Given two permutations P_{loop} and P'_{loop}, as defined above, with parameters a_0, a_1, a_2, a_3, F, and G, and a Clos network with parameters m (m even) and $r > 3$, then regardless of how the Clos network is programmed, $P_M(w, a_2)$ under permutation P_{loop} is not equal to $P_M(z, a_2)$ under permutation P'_{loop} for all $w \in F(a_2)$ and $z \in G'(a'_2)$.

Proof: Because of the swapping of certain parts of F, G, F', and G' at a_1 and a_3, to create permutation P'_{loop}, the sets $F'(a_2) = G(a_2)$ and $G'(a_2) = F(a_2)$. Since $P_{\text{loop}}(0,i) = P'_{\text{loop}}(0,i)$ for $0 \le i < m$ and $F(0) = F'(0)$ then $S(0) = S'(0)$. Therefore in both P_{loop} and P'_{loop} all of the inputs in F and F' go through the same switches. Since $F'(a_2) = G(a_2)$ and $G'(a_2) = F(a_2)$ and since $P_{\text{loop}}(a2, x) = P'_{\text{loop}}(a2, x)$ for $0 \le x < m$ the maps $P_M(w, a_2) \neq P_M(z, a_2)$ for $w \in F(a_2)$ and $z \in G'(a_2)$. ||

Using this theorem, dependencies between any output $P_M(z, a_2)$ for $z \in F(a_2)$ and all of the inputs $I(a_1, x)$ and $I(a_3, y)$; $x \in F(a_1)$ and $y \in F(a_3)$ can be shown. By applying the theorem many times it is possible to show the dependency between an output and many individual inputs (as opposed to being dependent upon a set of inputs).

Theorem 6: There exist outputs of any DAG to program the center stage switches of a Clos network with parameters m (m even) and $r > 3$ which are dependent upon at least $(m/2+1)(r-3)$ inputs.

Proof: Call an output to which the theorem refers $P_M(x,y)$, $(y \neq 0)$. For each i and j $0 < i < j < r$ where $i \neq y$ and $j \neq y$ construct a permutation P_{loop} with a fixed loop of length 4 and with $a_0 = 0$, $a_1 = i$, $a_2 = y$, and $a_3 = j$, and arbitrary sets f and g, and arbitrary b_i. For each of these permutations construct a permutation with a rearranged loop in the fashion of P'_{loop}. By Theorem (5) and since all pairs of i and j are selected $P_M(x,y)$ depends upon every possible pair of input sets except input stage 0 and input stage switch y (an input set is a set of inputs to the programming hardware which all map inputs to the Clos network entering the same input stage switch), then by Theorem (2), output $P_M(x,y)$ depends upon $(r-3)$ of the input sets.

For each of the permutations described above, generate $\binom{m}{m/2}$ additional permutations, one permutation for each possible permutation of those inputs mapped to f. For each of these a permutation like P'_{loop} can be created. By Theorem (1) then each output depends upon at least $(r-3)(m/2+1)$ inputs. ||

Corollary 6.1: For $r > 3$, the programming hardware for a Clos network using devices with fan-in of w will require at least $O(\log_w (r-3)(n/2+1))$ path-length. ||

The result just derived does not work with Clos networks of type $r < 4$. To find dependencies in these networks the following theorem will be used:

Theorem 7: An output $P_I(x,y)$ $(x > 0)$ depends upon at least $m-1$ inputs.

Proof: Construct a family of permutations P and P' in the following way:

$$P(0,i) = (0,i)$$

for $0 \leq i < z$

$$P(0,i) = (0,i-1)$$

for $z < i < m-1$.

$$P(0,z) = (1,0),$$

$$P(x,y) = (0,m-1),$$

with the remaining elements unspecified. P' is identical to P except for the following 2 elements:

$$P'(0,w) = (1,0) \quad \text{and} \quad P'(0,z) = (0,w),$$

where $0 \leq w < z < m$. With permutation P the only path from input stage switch x to output stage switch 0 is through center stage switch z. With permutation P' the only path from input stage switch x to output stage switch 0 is through center stage switch w. Therefore output $P_I(x,y)$ must be different under permutations P and P'. Since x and w can take on all pairs of values then by Theorem (1) $P_I(x,y)$ depends upon at least $m-1$ inputs. ||

Corollary 7.1: Any programming hardware for a Benes network [1], for which $m = 2$ $r = n/2$,, and $w = 2$ will require a path of at least length $O(\log_2(n-6))$ (assuming $n > 6$) path-length. Likewise, any programming hardware for a complementary Beneš network [2], which has parameters $m = n/2$ and $r = 2$, will require a path of at least length $O(\log_2(n/2-1))$. ||

4. Programming Complexity of Recursively Decomposable Permutation Networks

In the earlier section, we derived lower bounds for the programming time of a 3-stage Clos network. In this section, we examine the programming complexity of a recursively decomposable 3-stage network.

The programming hardware for this type of network will be designed to be naturally recursive; its input will be a permutation to be realized, its output will be a permutation to be realized by the smaller networks which make up the large one and switch settings for the remainder of the network. The programming hardware, then, only need be described for an arbitrary level in the recursion and for a small version of the network (i.e., the 2 input case), its efficacy for any size network can then be inductively determined using a generic programming algorithm as shown below.

```
1    Algorithm set-up($n, P$)
2    Begin
3        if $n > 1$ then
4        begin
5            $R \leftarrow$ set-up-level($n, P$);
6            $P' \leftarrow$ make_smaller_P($n, P, i$);
7            $n' \leftarrow$ reduce_n($n, P, i$);
8            for $i \leftarrow 1$ to $k$ do begin ;
9                set-up($n', P'$) // Call Set-up;
10           end ($k$ loop);
11       end
12       else $R \leftarrow$ root_calc;
13   end (set-up);
```

The algorithm uses two variables as input,, namely, n and P. P is the input permutation and n is its size . In line 3 the size of the permutation is tested. If it is below some predetermined size (here 2) then some "calculations" are performed in line 12 by a function called root_calc and the algorithm ends. If $n > 1$ then the algorithm performs some calculations in lines 5,6 and 7, storing the result in R, a variable of some arbitrary type, and computing the new values for P and n. After the calculations for one level of the algorithm are performed the algorithm recursively calls itself several times (lines 10 through 11), each time with a smaller and different data set.

The execution time of a recursive programming algorithm as described above of size n is specified by the recurrence

$$\mathcal{T}(n) = k\mathcal{T}(\frac{n}{d}) + g(n), \tag{6}$$

where k and d are positive integers and $g(n)$ is the amount of time it takes to perform the calculations at a level of the recursion at which the data size is n. The integers k and d, and the function g are three important determinants of a recursive algorithm's complexity and make up. More precisely, k is inversely proportional to the amount of parallelism in the recursive algorithm, the rate of reduction of the size of the permutation to be programmed, and $g(n)$ is the execution time of the serial parts of the algorithm. Together these three parameters characterize a recursive algorithm.

What is more is that we can use the bounds which are obtained in the earlier section to compute lower bounds on the programming time of a recursively decomposed Clos network. The path-length or time bounds of the earlier section are valid for $g(n)$ since $g(n)$ denotes the time complexity of one level of a programming algorithm for a recursively decomposed Clos network. In addition to $g(n)$ we also must specify values for k and d in computing lower bounds with (6). In the table below, we list the time complexities for $d = 2$, small values of k and some typical complexities for function $g(n)$ that might be encountered in programming hardware for permutation networks.

Serial-	Time Complexity of Serial Part, $g(n)$				
ism (k)	$O(1)$	$O(\log n)$	$O(\log^2 n)$	$O(n)$	$O(n^2)$
k=1	$O(\log n)$	$O(\log^2 n)$	$O(\log^3 n)$	$O(n)$	$O(n^2)$
k=2	$O(n)$	$O(n)$	$O(n)$	$O(n \log n)$	$O(n^2)$
k=3	$O(n^{1.59})$	$O(n^{1.59})$	$O(n^{1.59})$	$O(n^2)$	$O(n^2)$

Values of $\mathcal{T}_n$ for $d = 2$, and some values of k and $g(n)$.

Various complexities for $g(n)$ head the columns while values of k label the rows. At an intersection is the time complexity of an algorithm with k equal to the row label and $g(n)$ equal to the column head. In particular, the entries in the second column are lower bounds for the programming time of a recursively decomposed Clos network for small values of k.

The table also reveals a few unexpected complexities. For example, the amount of parallelism introduced does not affect the time complexity of an algorithm with $g(n) = n^2$. Furthermore, the time complexity of an algorithm with $k = 2$ or $k = 3$ and $g(n) = O(1)$ is the same as such an algorithm with $g(n) = O(\log n)$ and $g(n) = O(\log^2 n)$. It is also revealed that a recursive algorithm which has $O(n^2)$ execution time at each level will not benefit from parallelism, and with some serialism (i.e., $k \neq 1$) having a time complexity at one level of $O(\log^2 n)$ does not perform slower (except for a constant factor) than one with a time complexity of $O(1)$ at a level. On the other hand, when $k = 1$, i.e., with the highest degree of parallelism , each improvement in time complexity at a level improves the overall time complexity.

It is possible to generalize these expressions in the table for arbitrary values of d and k. These generalizations and their discussion will be deferred to another place.

5. Concluding Remarks

This paper examined the parallel time complexity of programming permutation networks. Lower time bounds have been derived for 3-stage and recursively decomposed Clos networks using a directed acyclic graph model of dependency. It is worthwhile to point out that dependencies among permutations that are derived in Section 3 remain valid even if the directed acyclic graph structure of the dependency model is relaxed. The directed of acyclic graph model with restricted fanin is used just for deriving the logarithmic lower bounds. In a fuller version of this paper, we shall examine how our bounds behave with other models of parallel computation.

References

[1] V.E. Beneš , "Optimal rearrangeable multistage connecting networks," *The Bell System Technical Journal,* pp.1641–1656, July 1964.

[2] J.D. Carpinelli and A. Y. Oruç, "Applications of edge-coloring algorithms to routing in parallel computers," *Proc. of 3rd International Conference on Supercomputing,* May 1988.

[3] C. Clos, "A study of non–blocking switching networks," *The Bell System Technical Journal,* pp. 406–424, March 1953.

[4] T-Y Feng, "A Survey of Interconnection Networks," *IEEE Computer*, December 1981, pp. 12–27.

[5] G.F. Lev, N. Pippenger, and L.G. Valiant "A Fast Parallel Algorithm for Routing in Permutation Networks," *IEEE Transactions on Computers*, Vol. C–30, No. 2, February 1981, pp. 93–100.

[6] G.M. Masson, G.C. Gingher, S. Nakamura, "A Sampler of Circuit Switching Networks," *IEEE Computer*, June 1979, pp. 32–48.

[7] D. Nassimi and S. Sahni, "Parallel Permutation and Sorting Algorithms and a New Generalized Connection Network," *Journal of the ACM*, Vol. 29, No. 3, July 1982, pp. 642–667.

[8] D. Nassimi and S. Sahni, "Parallel Algorithms to Set Up the Beneš Permutation Network," *IEEE Transactions on Computers*, Vol. C-31, No. 2, February 1982, pp. 148–154.

[9] D.C. Opferman & N.T. Tsao–Wu, "On a Class of Rearrangeable Switching Networks," *Bell System Technical Journal*, May-June 1971, pp. 1579-1600.

[10] A. Yavuz Oruç, "Rearrangeable networks based on double coset decompositions of symmetric groups," 21st Annual Conference of Information Sciences and Systems, John Hopkins University, Baltimore MD, pp. 453–458.

[11] A. Waksman, "A permutation network," *Journal of the ACM*, vol. 15, No. 1, January 1968, pp. 159–163.

LIMITED WAITING FOR VALUABLE RESOURCES: AN ADAPTIVE CONTROL STRATEGY FOR CIRCUIT-SWITCHED NETWORKS

GITA GOPAL and ABEL WEINRIB
Bellcore, 445 South Street, Morristown, NJ 07960-1910

ABSTRACT

We present an adaptive control strategy for circuit-switched networks in which calls are permitted to wait for a limited time to be carried on a low-cost route. The strategy uses network resources more efficiently than other conventional approaches, exhibiting lower blocking of calls under overload conditions. Our strategy is also adaptive in that when there is little network congestion, there is no penalty for our policy, in contrast to other schemes in which improved performance under overload is achieved at the cost of degraded performance at lower loads.

1. INTRODUCTION

The strategy employed in a circuit-switched network to allocate resources for connection requests has a major impact on the network performance and on the total number of physical resources required to support a given traffic load. This resource allocation problem has been widely studied in the context of telecommunications networks, leading to a variety of routing strategies to decide between the many possible source-destination routes for an incoming call. These routing schemes include static policies in which a set of routes is tested in a predetermined order [Ash81], dynamic policies in which the state of the network is examined to determine the desired order of routes for a particular call [Syzb79, Akse85], and policies which explicitly reserve capacity for calls using low-cost preferred routes over calls using higher-cost alternate routes [Krup82]. These schemes exhibit different behavior in terms of *network blocking*, the percentage of calls unable to obtain network resources, particularly in the presence of *network overload*. Network overload may be caused by a uniform growth in traffic patterns, or may be due to transient increases which are the result of natural disasters, Bruce Springsteen concerts, etc. Some schemes are also more adaptable than others to these fluctuations in the offered load.

In this paper, we present an adaptive control strategy for circuit-switched networks in which calls are permitted to wait for a limited time to be carried on a low-cost route. If the route becomes available before the timer expires, the call is carried on that route. If the route is still not available at the end of this limited waiting period, alternate routes are attempted in some order, e.g. using the dynamic policy mentioned earlier. During periods of high overload, our policy exhibits much lower network blocking than other schemes. In addition, our routing policy is adaptive in that when there is little network congestion, the waiting interval is seldom utilized and there is no penalty for our policy, in contrast to other schemes in which improved performance under overload is achieved at the cost of degraded performance at lower loads.

In section 2, we define the routing problem for circuit-switched networks and describe in detail the three policies suggested, and in some cases implemented, in the past. In section 3, we describe our new routing policy based on limited waiting. In section 4, we compare the performance of our scheme to the other schemes, using a simulation model of a network based on the Chicago Metropolitan area.

2. *ROUTING IN CIRCUIT-SWITCHED NETWORKS*

In a circuit-switched network (see Figure 1), such as the public telecommunications network, an incoming call arrives at the source switch A and must obtain a connection to the destination switch B within an acceptable call set-up interval, or be abandoned (blocked). In our model, all calls are given equal priority. An incoming call can be carried on many possible routes: it may utilize a trunk on the trunk-group directly connecting switches A and B, or the call may be carried on a multi-hop route via other intermediate switches. The goal of the routing problem is to maximize overall network performance over time, i.e. minimize percentage blocking, by choosing a route for a call that minimizes interference to other calls that may arrive in the future. Another measure of interest is the average call setup delay due to the routing policy; this delay should be within some acceptable bound, e.g. in the range of a few seconds.

Finding the optimal policy that achieves minimum blocking is intractable, but heuristics such as "shortest route" and "least-loaded route" have been suggested in the past. In general, a multi-hop route has a substantially higher cost than a direct route; a call carried on a multi-hop route can potentially block a number of other calls that might use the same trunks as direct routes. For this reason, networks are generally engineered to permit calls to only use the direct route and some small set of two-hop routes [Ash81]. Calls that cannot be carried on any route from this set are blocked. For the same reason, networks tend to experience less overall blocking if as many calls as possible are carried on the direct route.

If the direct route is unavailable, using the least-loaded two-hop route is a good heuristic. This policy spreads the load, making it less likely that a two-hop call will use up the last trunk that is the direct route for some other call or that is some other call's only possible route. These heuristics have been incorporated in routing strategies in the past.

Static routing [Ash81], based on estimated traffic forecasts, has been implemented in the telephone network. Different routes are tested in a fixed order, starting with the direct route. An available trunk originating at the switch is seized and a request to allocate the rest of the route is sent to the next switch along the route. If the route being tested is not completely available, the seized resources are released, and another route is tested. The order of the routes is predetermined based on the estimated traffic; this scheme does not take into account information about the network state at the time of the call attempt, other than that some trunks have no free capacity.

Other more sophisticated schemes have been suggested in which some global information is maintained about the state of the network. These *least-loaded routing* (LLR) schemes setup a call by first testing the direct route, and then testing the other multi-hop routes ordered according to decreasing free capacity as determined by global state information. The state information may be maintained either centrally [Szyb79], or in a distributed manner [Akse85]. LLR schemes are more adaptable to fluctuations in traffic, since network state is used in the call set-up decision. In fact, we have found that the dynamic LLR scheme outperforms the static routing scheme at all levels of overload [Wein87]. We will therefore omit the static scheme in further discussions in this paper.

The performance of both static and dynamic schemes suffers under overload because they carry too many two-hop overflow calls; two-hop calls use up capacity that could be better used by directly routed calls [Akin84]. The schemes' performance can be improved by implementing *trunk reservation* [Krup82]; we shall denote the adaptive LLR scheme combined with trunk reservation by TR. Under trunk reservation, each trunk group in the network has a predetermined trunk-reservation level l, which may either be a fixed number of trunks for every trunk group or based on a fraction f of the trunk group's total capacity, in which case $l = \lceil f * capacity \rceil$. When a call arrives, the switch attempts to set it up on the preferred route, which is usually the direct route. (If the network is not fully connected, the preferred route may consist of two hops). A call using the preferred route is not subject to trunk reservation, and can seize even the last trunk in the trunk group. If the preferred route is not available, the *overflow* call can use an alternate route only if there are more than l trunks free on every hop. With TR routing, the choice of alternate route is based on remaining free capacity as in LLR, but the free capacity seen by overflow calls excludes the l reserved trunks.

By requiring that at least l trunks of a trunk group be free before an overflow call can use one of the trunks, trunk reservation gives priority to directly routed calls and improves network performance at high overloads. However, a drawback of trunk reservation is that at low loads it hurts the performance, since it rejects overflow calls that should in fact be carried. Thus, for best performance trunk reservation needs to be "turned on" only during network overload, leading to additional network control complexity.

3. *THE LIMITED WAITING POLICY*

We now present our *limited waiting* scheme (LW), a different approach to giving priority to calls that use a preferred route. Our scheme does not have the drawback of explicitly reserving capacity for such calls, and thus can be continuously in force. In fact, the LW policy exhibits better performance than LLR for all loads, although it is at higher loads that the improvements are most pronounced. This improvement in network operations is gained at the cost of only slight changes to the call setup procedure.

As the hardware used for setting up calls in circuit-switched networks (e.g. switch processors and control channels) becomes more powerful, the setup procedure can be made ever faster. However, as long as the calls are being set up for human users, as in the telecommunications network, shortening the call setup delay below a few seconds becomes unobservable and irrelevant to the user. In the LW scheme, the extra time available for setup is used to run the network more efficiently. In fact, the results presented in Section 4 indicate the potential to greatly improve network performance by increasing the call setup period by small amounts, subject to user tolerance constraints.

In the LW scheme, the extra time available for call setup is used to queue the call for the preferred route, if this route is unavailable at call arrival. If the preferred route becomes available within some limited waiting interval T, the call uses this route. Otherwise, at the end of the interval T the call is set up using the standard LLR algorithm on alternate routes (i.e. without any queueing). When the preferred route consists of two hops, both of which are unavailable at call arrival, the call is entered in both queues simultaneously; thus the queueing delay is still at most T. If either hop is available, a trunk on that hop is held while the call is queued for the other hop.

The difference between our scheme and the standard LLR approach is that we give extra priority to the preferred route for a call, by allowing it to wait a time T for this route to become available before trying for the alternate two-hop routes. When the network is not very congested, this approach adds little or no delay to the call setup time, since the preferred route will either be available at call arrival, or become available shortly thereafter. Thus, a call will rarely have to wait the full interval T.

To gain insight into the behavior of the LW policy, consider a simple model consisting of a single trunk group that is offered primarily traffic that can queue for up to a time T (preferred traffic) and a negligible amount of traffic that can not queue (overflow traffic). With Poisson arrivals and exponential service times and a trunk group of size m, this model is equivalent to an M/M/m queue with customers that leave the queue after a fixed time T; this model has been solved exactly [Gned68]. In Figure 2 we show the blocking of preferred and overflow traffic as a function of T for various trunk group sizes. The primary effect of allowing preferred calls to wait up to a time T is that the blocking of overflow calls increases significantly while the blocking of preferred calls decreases slightly. Thus, the LW scheme automatically gives priority to preferred calls over overflow calls. The overflow call gets to use a trunk group only if it has available trunks and if the associated queue is empty. Under these circumstances the probability of an overflow call blocking a direct call is small, since no direct call has arrived in the past T seconds without obtaining a trunk. As we shall see in the next section, blocking one overflow call allows the network to potentially carry more than one preferred call, which leads to greatly improved network performance in terms of blocking.

We should emphasize that our LW scheme is different from other call setup techniques involving queueing calls. Previously proposed schemes involve queuing a call only after the call has failed on *all* of the routes, both preferred and overflow (e.g. see [Gira85]). These schemes are very different from our LW scheme that queues a call first for a preferred route, and only then tries for the overflow routes. Queueing a call after it has tried all of the routes does not give priority to direct calls, and thus will not exhibit the improved performance under overload of our scheme.

4. RESULTS

Since routing strategies in telecommunications networks are too complex to study analytically in a realistic context, we have built a simulation model to study their behavior. We simulate call-by-call traffic flow and compare the results of our limited waiting policy with simulations of traffic flow using LLR and LLR with trunk reservation (TR). The experiments described are based on a 38-node network model of the Chicago metropolitan area, although similar results have been obtained on other networks. The Chicago network topology has been used before to study other schemes (e.g. [Fiel83, Ott85]). The nodes in the network are connected by a total of 586 trunk groups; some of the preferred routes are therefore two-hop routes. The call arrival distribution is Poisson, with an overall arrival rate at design load (summed over all source-destination pairs) of 44 calls per second, and the call holding time distribution is exponential, with an average call holding time of 180 seconds. The network was engineered using the network design algorithm proposed by Ash et al. [Ash81]. Calls between a source-destination pair can be routed on either direct or two-hop paths, and all three schemes (LLR, TR and LW) use the same set of admissible paths. The graphs displayed below are the averaged results from several busy hours of traffic with the initial data discarded to preclude transient conditions. We first discuss the network blocking achieved by the various policies, and then comment on the average call setup delay seen by the user.

Figure 3 shows network blocking for the LW scheme as a function of the limited waiting interval T, for different levels of network overload. Network overload is expressed as a multiple of engineered load (1.0 corresponds to the engineered load). Note that when $T=0$, the LW scheme reduces to the standard LLR scheme. Observe that even limiting the waiting interval to be at most 3 seconds drastically reduces the network blocking. For example, at 20% overload, the LLR scheme leads to over 11% blocking, while the LW scheme with $T=3$ shows less than 7% blocking. If T can be increased to say 10 seconds, the blocking can be reduced to around 5%, which is less than half the blocking experienced with LLR. When the network is less loaded, the LW policy has less noticeable impact on network blocking. However, there is never a penalty to using this policy; network blocking is always lower with the LW policy than with LLR, and decreases with increasing T.

So far, we have compared LW to LLR and shown that LW exhibits uniformly lower network blocking for different levels of overload. Before we compare LW to the trunk reservation scheme TR, we first present some results to provide insight into the effect of the trunk reservation level l on the behavior of TR. In Figure 4, we plot network blocking versus different trunk reservation fractions for several different values of network overload. Note that the limit $f \rightarrow 0^+$ is equivalent to a uniform trunk reservation level $l=1$. The blocking of the LLR policy is displayed for comparison. At high overloads, small trunk reservation fractions (or even only one reserved trunk) can lead to a large reduction in network blocking. While this large reduction due to reserving just one trunk may seem counter-intuitive, it may be explained as follows. When a trunk group is operating at full capacity and a call terminates, the released trunk is reserved for preferred traffic, and during high overload, it is likely that a call will arrive in the near future and seize the reserved trunk. The effect is therefore that most released trunks are reserved for preferred traffic, even though the reservation level is set at only one trunk. At low overloads however, trunk reservation can actually increase the blocking, as seen in Figure 4, because reserved trunks may remain idle waiting for preferred traffic while overflow calls are unnecessarily blocked.

There is thus a cost to trunk reservation which is made clear in Figure 5. In this figure, we plot blocking versus overload for various schemes, LLR, LW with $T=1$, 5 and 10 seconds, and TR with a trunk reservation fraction of 3%. This trunk reservation level was chosen from the results of Figure 4 as it gives good performance at high overloads without too much penalty at low overloads. It can be seen from Figure 5 that, with a preset trunk reservation level, the TR scheme actually performs *worse* than the LLR scheme for overloads up to 1.08. Therefore, while TR helps in periods of high overload, there is a penalty during normal network operation or during periods of low overload. On the other hand, the LW scheme never suffers a penalty. With a preset waiting time T, the scheme will perform at least as well as LLR at low overload, with the potential to do much better at high overloads. In fact, with $T=5$, the LW scheme does better than the TR scheme with any trunk reservation level, for the range of overloads considered.

We now discuss the effect on call setup delay due to the LW policy. In the simulation we assumed instantaneous setup for LLR and TR to isolate the effect of queueing for trunks; in practice the call setup delay for LLR and TR can take on the order of a second. The LW policy increases the call setup delay by a small amount. In fact, when $T=5$ seconds, the average increase in call setup delay due to the LW policy is 0.45 seconds when the network is operating at engineered loads and 1.40 seconds when the network is 20% overloaded. This demonstrates the adaptive nature of the LW policy: calls are forced to wait only when required. The small increase in average call setup delay should be acceptable to the human user.

5. *DISCUSSION*

In this paper we have presented an adaptive control strategy for the important problem of resource allocation for connection requests in a circuit-switched network. The LW policy allows calls to wait for a limited time for valuable resources before attempting the standard call setup policy. During periods of high overload, this has the effect of maintaining good network behavior, by allowing trunks to be used by preferred calls rather than overflow calls. When the network is operating normally, or at low overload, there is no penalty to the scheme in terms of network blocking. This improvement in network operations is gained at the cost of only slight changes to the call setup procedure, and with relatively small increases in average call setup delay.

REFERENCES

[Akin84] J. M. Akinpelu "The Overload Performance of Engineered Networks with Nonhierarchical and Hierarchical Routing", Bell Labs Technical Journal, Vol. 63, p. 1261, 1984.

[Akse85] B. Akselrod and G. Langholz "A Simulation Study of Advanced Routing Methods in a Multipriority Telephone Network", IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-15, No. 6, 1985.

[Ash81] G.R. Ash, R.H. Cardwell, and R.P. Murray "Design and Optimization of Networks with Dynamic Routing", Bell System Technical Journal, Vol. 60, pp 1787-1820, 1981.

[Fiel83] F.A. Field "The Benefits of Dynamic Nonhierarchical Routing in Metropolitan Traffic Networks", Proc. Tenth Int. Teletraffic Congress, Montreal, 1983.

[Gira85] A. Girard and V.G. Taillon "Performance Evaluation of Private Networks with Automatic Route Selection and Call Queueing", Proc. Eleventh Int. Teletraffic Congress, Kyoto, 1985.

[Gned68] B.V. Gnedenko and I.N. Kovalenko *Introduction to Queueing Theory*, p. 33, Israel Program for Scientific Translations, Jerusalem, 1968.

[Krup82] R.S. Krupp "Stabilization of Alternate Routing Networks", Proc. IEEE ICC, Philadelphia, 1982.

[Ott85] T.J. Ott and K.R. Krishnan "State Dependent Routing of Telephone Traffic and the Use of Separable Routing Schemes", Proc. Eleventh Int. Teletraffic Congress, Kyoto, 1985.

[Syzb79] E. Szybicki and A.E.Bean "Routing in Local Telephone Networks: Performance of Proposed Call Routing Algorithms", Proc. Ninth Int. Teletraffic Congress, Torremolinos, 1979.

[Wein87] A. Weinrib and G. Gopal "Decentralized Resource Allocation for Distributed Systems", Proc. IEEE Infocom, San Francisco, 1987.

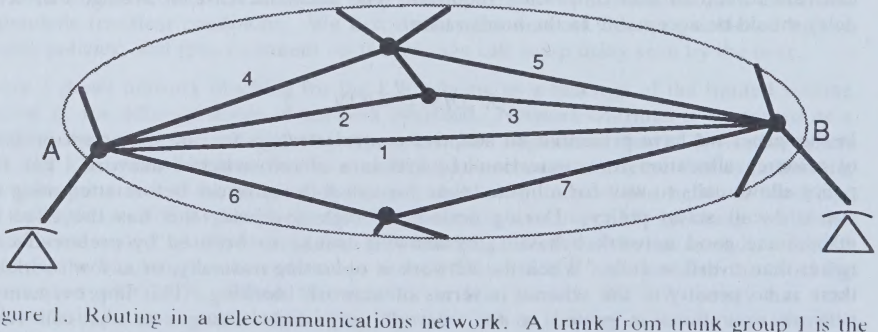


Figure 1. Routing in a telecommunications network. A trunk from trunk group 1 is the direct route, while one trunk from each of groups 2 and 3 form a two-hop overflow route.

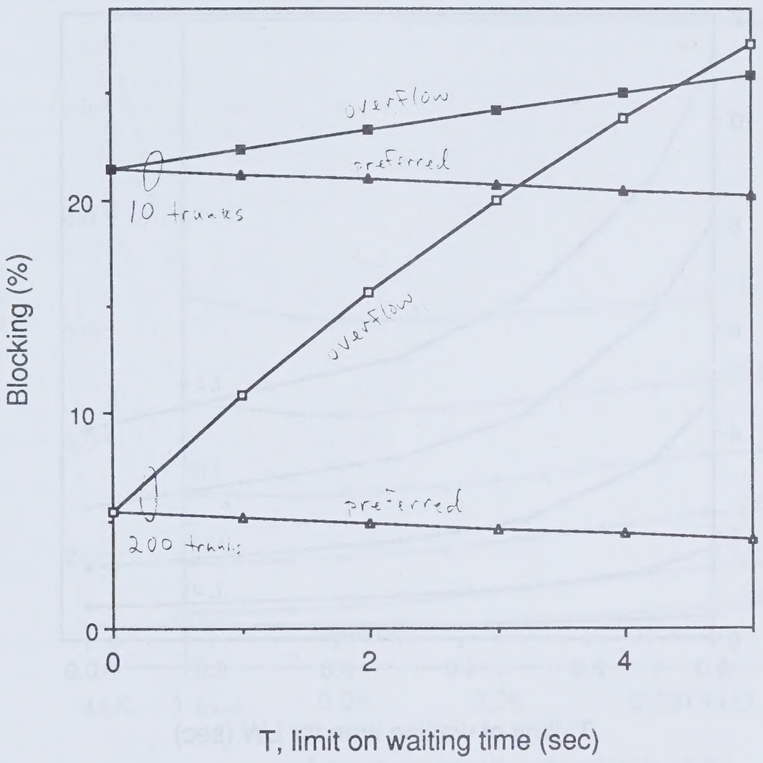


Figure 2. Blocking against the limit of waiting time (T) for a single trunk group subject to primarily preferred traffic and a small amount of overflow traffic. The load is set equal to the capacity of the trunk group, and the mean holding time is 180 seconds. Curves for trunk groups of size 10 and 200 are displayed. In each case, the upper curve is the blocking of overflow calls, and the lower curve is the blocking of preferred calls.

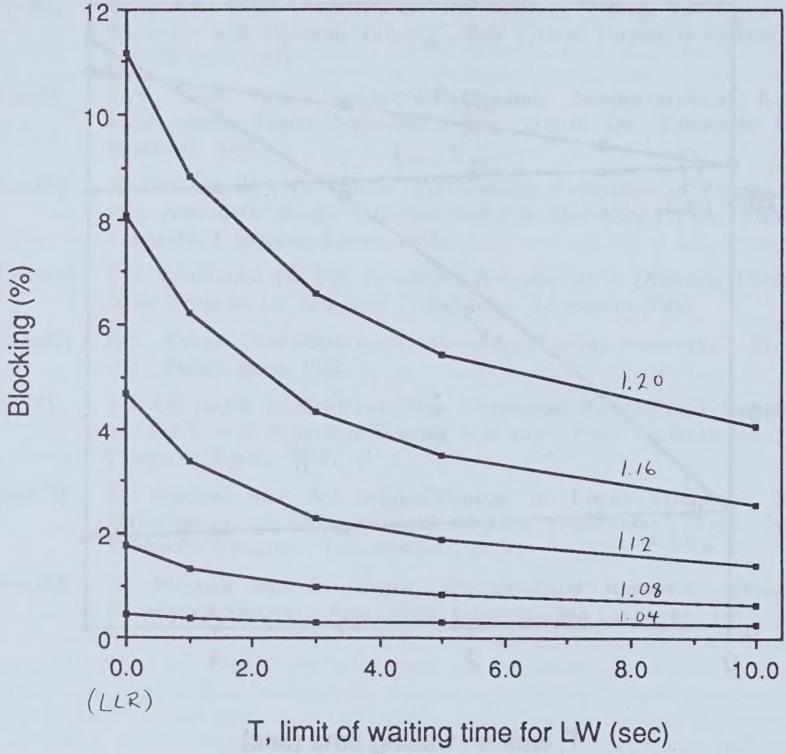


Figure 3. Blocking against the limit of waiting time (T) for the LW policy at various overloads (1.04, 1.08, 1.12, 1.16, and 1.20). The LW policy with $T=0$ is equivalent to the LLR policy.

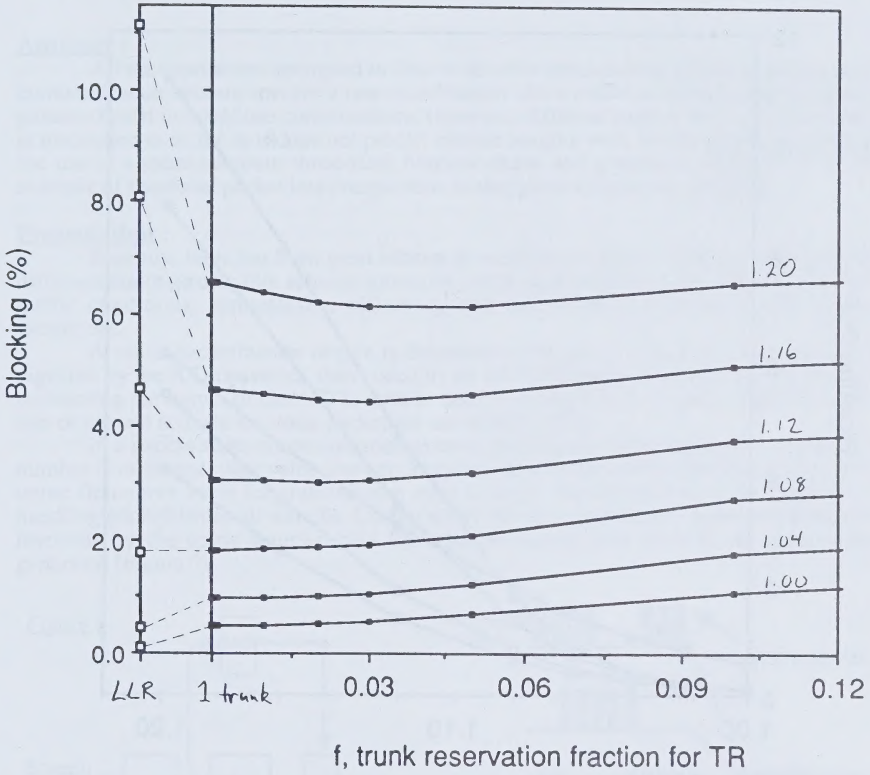


Figure 4. Blocking against trunk reservation fraction (f) for the TR policy at various overloads (1.00, 1.04, 1.08, 1.12, 1.16, and 1.20). The limit $f \to 0$ is equivalent to a uniform trunk reservation level $l=1$. The blocking for the LLR policy is displayed at the left.

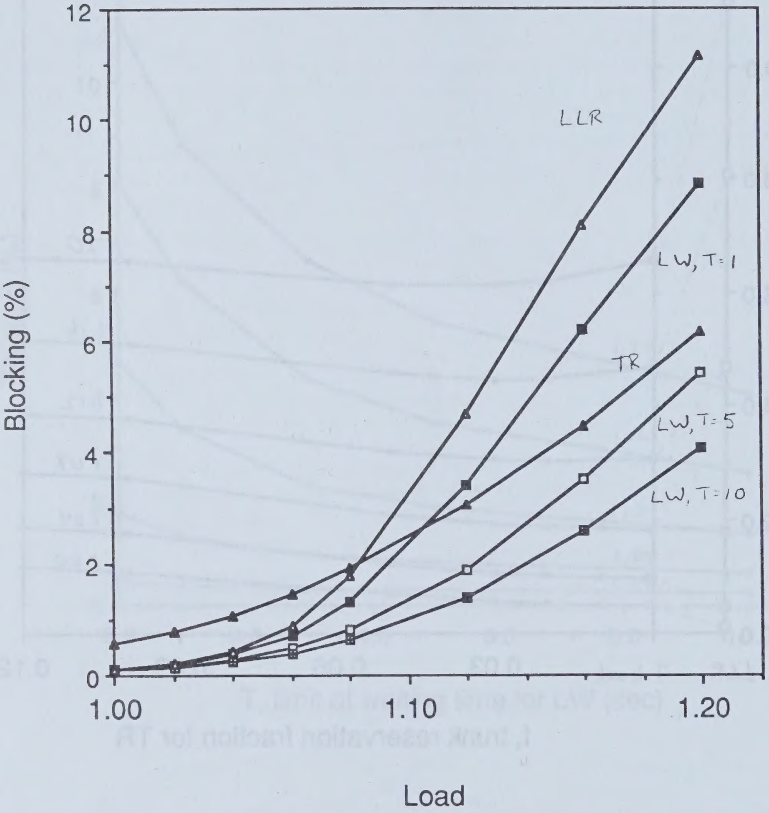


Figure 5. Blocking against load for the three policies: LLR; LW with $T=1$, 5, and 10 seconds; and TR with a trunk reservation fraction of $f=0.03$.

Voice Transmission in Packet Switching Networks : A Model for Queueing Analysis

Frank M. Brochin and **John B. Thomas**
Department of Electrical Engineering
Princeton University, Princeton, NJ, 08544

Abstract :

All the approaches attempted to date to describe the queueing effects in packet voice communication systems involve a two-state Markov chain model to characterize the speech patterns found in telephone conversations. However, different studies show that this model is inaccurate in so far as it does not predict silence lengths well. In this paper, we propose the use of a more accurate three-state Markov chain and present a solution for it. An example of fractional packet loss computation is also given to illustrate the model.

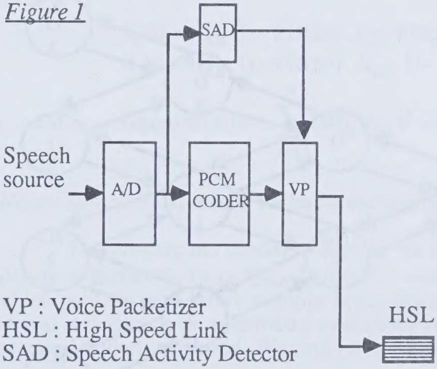
Presentation :

Recently, there has been great interest in voice transmission in packet form. Indeed, different factors make this scheme attractive, such as adaptability to changing network traffic conditions, transmission efficiency and the possibility of using new coding techniques.

A voice packetization device is described in Figure 1. The speech signal is first digitized by the A/D converter, then coded by an ADPCM coder. A speech activity detector monitoring the output of the A/D is used to decide whether there is speech activity on the line or not and activate the voice packetizer accordingly [3, 9].

In a packet voice communication system, packets are fed to a common queue by a number N of independent voice sources. They are then removed from this queue on a first-come first-serve basis for transmission over a single fast channel of finite capacity C, handling packets from all sources. During active periods, packets are generated at regular intervals by the voice sources (rate R), whereas during idle periods, no packets are generated (Figure 2).

Figure 1



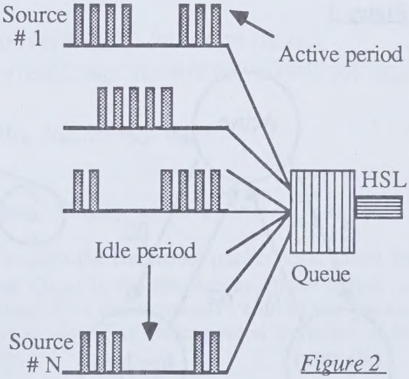


Figure 2

Due to psychological effects, and because of the real time reconstitution process that discards packets arriving too late, the delivery time of voice packets must have a small mean, a small variance and cannot exceed a certain threshold. Because of the switching scheme used, a fundamental problem to study is the delay performance in the statistical multiplexer at the entrance to the network [9].

Related work :

Studies by Brady [6-7] and Minoli [8] have shown that the speech patterns in each voice source can be represented by a finite Markov chain. Because of the complexity of the problem, and for simplification purposes, the approaches attempted to date have used a two-state Markov chain model, yielding the packet delay distribution for the packets in the multiplexer [1-5]. However, Brady and Minoli showed that this two-state Markov chain could not represent accurately the silence length of a speech pattern as found in a telephone conversation, since the model did not match empirical data. A more general model was recently proposed [10], but no solution was given for the delay probability distribution.

Our model and solution :

We propose here a three-state Markov chain (Figure 3), two states representing short and long silence periods, one state representing an active period. Because of the three possible states for each voice source, we define the phase process as the continuous time Markov chain whose state J represents the number of speakers that are active,"idle short", or "idle long" (Figure 4).

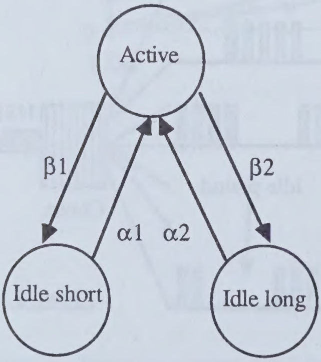
We find the probability distribution for the delay by considering an imbedded Markov chain to represent the queueing process. This Markov chain is defined at the instants t_n when the number of active voice sources steps from below the capacity of the link (underload) to a value above this capacity (overload). If we denote by $Q(t_n)=Q_n$ the queue length at time t_n, and by U_n the net queue length change that would occur if the queue never emptied during the n^{th} overload/underload cycle, then [1] : $Q_{n+1} = \max(0, Q_n + U_n)$. Defining :

$$G(z) = \sum_{q=0}^{\infty} P(Q \le q) \,.\, z^q \qquad H(z) = \sum_{u=-\infty}^{\infty} P(U=u) \,.\, z^u$$

(where we assumed ergodic probabilities), we get $G(z) = [H(z) \,.\, G(z)]_+$ where $[.]_+$ denotes the truncation of its argument to include only non-negative powers of z .

A method for solving this equation, once H(z) is known, was develloped in [1]. H(z) is easily found, based on the knowledge of the $H_i(z)$, the generating functions of the distributions of the virtual queue increment, when J=i (state of the phase process).

Figure 3



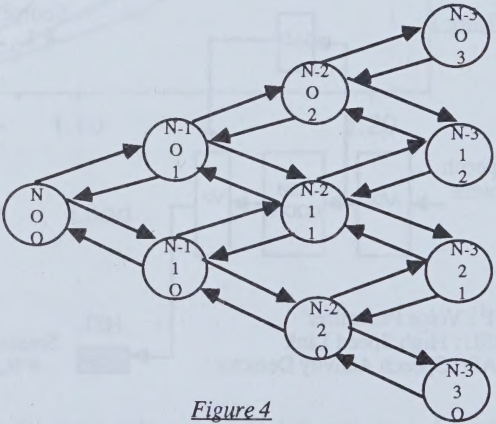


Figure 4

For our model, if we denote by t the length of stay in state i, then :

$$P\{U=n, n\geq 0 \mid N-i > C\} = P\{n \leq R(N-i-C).\ t < n+1\} = \int_{n/R(N-i-C)}^{(n+1)/R(N-i-C)} \lambda_i e^{-\lambda_i t}\, dt = a_i^n.(1-a_i)$$

$$P\{U=-n, n\geq 0 \mid N-i < C\} = b_i^n\ .\ (1-b_i) \quad \text{with} \quad a_i = \exp(-\lambda_i / R(N-i-C)) \quad \text{and} \quad b_i = a_i^{-1}$$

λ_i represents the sum of all the rates coming out of state i. We have here :

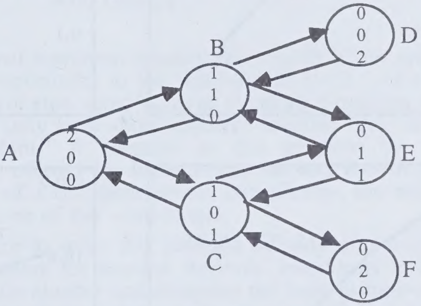
$\lambda_i = (N-i).(\beta_1+\beta_2) + i.\alpha_1$ (states of the type (N–i, i, 0))
$\lambda_i = (N-i).(\beta_1+\beta_2) + i.\alpha_2$ (states of the type (N–i, 0, i))
$\lambda_i = (N-i).(\beta_1+\beta_2) + (i-k).\alpha_2 + k.\alpha_1$ (states of the type (N–i, k, i–k))

This enables us to compute the $H_i(z)$ and from there, $H(z)$.

Fractional packet loss :

To compare our results to those in [1], we consider the case when N=2 (Figure 5).

Figure 5



Defining : $H_{AB} = P_{AB}\ .\ H_A(z)$ (same notation for the other states), where P_{AB} is the probability of going to B while in A, we may compute H(z) by distinguishing between the paths starting at B and ending at B ($H_1(z)$), starting at B and ending at C ($H_2(z)$), starting at C and ending at C ($H_3(z)$), starting at C and ending at B ($H_4(z)$). An analysis of the phase process diagram shows that :

$$H_1(z) = H_{AB}\ .\ [1+\Psi(B,D)]\ .\ [1+\Psi(E,B)\ .\ (1+\Psi(E,C)\ .\ (1+\Psi(C,F)\ ))]\ .\ H_{BA}$$

$$H_2(z) = H_{AB}\ .\ [1+\Psi(B,D)]\ .\ H_{BE}\ .\ [1+\Psi(E,B)]\ .\ H_{EC}\ .\ [1+\Psi(E,C)]\ .\ [1+\Psi(C,F)]\ .\ H_{CA}$$

$$\text{where } \Psi(A,B) = \sum_{n=1}^{\infty} (H_{AB}.H_{BA})^n = H_{AB}\ .\ H_{BA}/(1-H_{AB}\ .\ H_{BA})$$

We get symmetrical results for paths starting from C. Finally, we have : $H(z) = \sum_{i=1}^{4} H_i(z)$

To compare our model to Stern's, we compute the fractional packet loss, given by ØLoss = $E(\max(0, Q_{ol} - Q_{max})) / L\cdot T$ where Q_{max} is the maximum queue length to which packets may arrive without being discarded, L is the expected value of the packet arrival rate during an overload / underload period, and T is the expected duration of an overload/underload period. We find for our model :

$$L = R\cdot\Pi_2\cdot(1+\Pi_2)$$

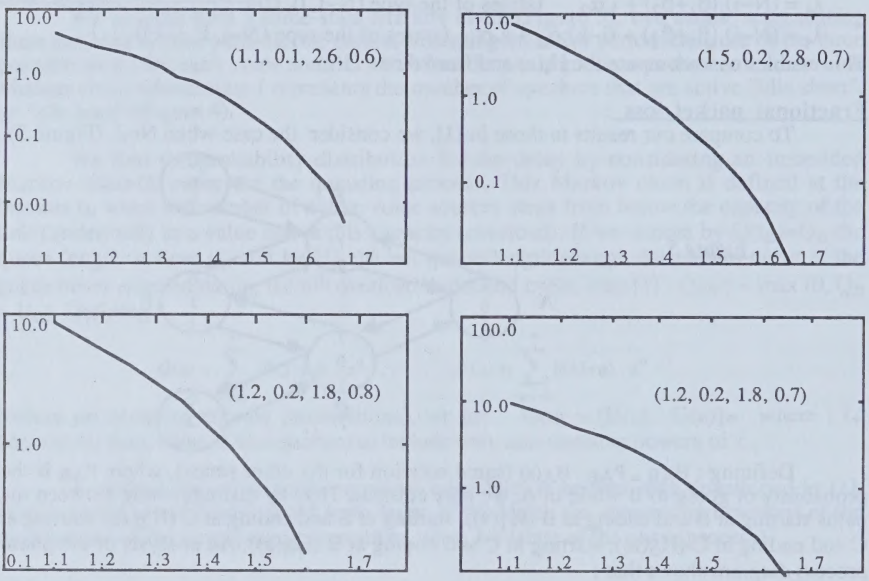
$$T = [2\cdot(\beta_1+\beta_2)\cdot\Pi_2^2]^{-1} \qquad \text{with } \Pi_2 = \alpha_1\cdot\alpha_2 / (\alpha_2\beta_1 + \alpha_2\alpha_1 + \alpha_1\beta_2)$$

Results are shown below for different values of the parameter set $(\alpha_1, \alpha_2, \beta_1, \beta_2)$. (R = 50 packets/s and we discard all the packets that are delayed by more than 0.2 s).

Conclusion :

In this paper, we present a more accurate representation for speech patterns, as found in telephone conversations. We also propose a solution for the queue length distribution in statistical multiplexers, as a generalisation to our model of Stern's method for a two-state Markov chain model. An example of fractional packet loss computation illustrates our model and solution.Results agree with previous studies [8], that showed that a two-state model predicts more traffic and loss than actual.

Fractional Packet Loss vs Capacity



Acknowledgements :

This work was supported by the U.S. Army Research Office under Grant DAAG-29-85-K-0191 and by the National Science Foundation under Grant ECS-87-13598.

References :

1. T.E. Stern, "A queueing analysis of packet voice" in *Proc. IEEE Global Telecomm. Conf.*, San Diego, CA, Dec. 1983, pp. 2.5.1-2.5.6.
2. H. Heffes and D. M. Lucantoni, "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance", *IEEE J. on Selected Areas in Communications*, No 6, September 1986, pp. 856-868.
3. Y.C. Jenq, "Approximations for packetized voice traffic in statistical multiplexers", Proc. IEEE Infocom, April 1984, pp. 256-259.
4. K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data", *IEEE J. on Selected Areas in Communications*, September 1986, pp. 833-845.
5. B. Kim, "Characterization of arrival statistics of multiplexed voice packets", *IEEE J. on Selected Areas in Communications*, December 1983, pp. 1133-1139.
6. P. Brady, "A statistical analysis of on-off patterns in 16 conversations", *Bell System Technical Journal*, vol 47, No 1, Jan. 1968, pp. 73-91.
7. P. Brady, "A model for generating on-off patterns in two-way conversations", *Bell System Technical Journal*, No 48, 1969, pp. 2445-2472.
8. D. Minoli, "Issues in packet voice communication", *Proc. of IEE.*, Vol 126, Aug. 1979. pp.729-741.
9. J.Turner and L. Wyatt, "A packet network architecture for integrated services", *Proc. Globecom 1983*, San Diego, CA, Nov. 1983, pp. 2.1.1-6.
10. V. Ramaswami and G. Latouche, "Modeling packet arrivals from asynchronous input lines", *Proc. of ITC*, Torino, Italy, June 1988, pp. 4.1A.5.1-7.

A NEW APPROACH TO SOLVE THE SEQUENCE-LENGTH CONSTRAINT PROBLEM IN CIRCULAR CONVOLUTION USING NUMBER THEORETIC TRANSFORM

HUIZHU LU
Dept. of Computing & Information Sciences
Oklahoma State University, Stillwater, OK 74078

SAMUEL C. LEE
Dept. of Electrical Engineering & Computer Sciences
University of Oklahoma, Norman, OK 73019

ABSTRACT

When the Mersenne or Fermat transform is used, the lengths of the sequences which can be convolved are linearly proportional to the word-length [1-3]. As a consequence, the one-dimensional circular convolution using number theoretic transform with modulo a Mersenne or Fermat number can only have short sequence lengths. This is known as the sequence-length constraint problem. A solution to this problem by using a two-dimensional implementation was offered by Rader, Agarwal and Burrus [2][4]. Using two-dimensional implementation of a one dimensional convolution, the maximum length of sequences equals twice the square of the word-length.

In this paper, a new approach to solve this problem by proposing a new formula to produce generalized modulo numbers for number theoretic transforms is described. By selecting a prime M as the modulo number and choosing the least primitive root of M as the α in the number theoretic transform and the radix m for hardware implementation, the sequence-lengths become exponentially proportional to the word-length and therefore the sequence-length constraint problem in one-dimensional convolution using number theoretic transform is solved.

The new formula generates a set of generalized modulo numbers including Mersenne and Fermat numbers. They provide not only more choices of modulo numbers for a number theoretic transform, but also simple implementation of fast circular convolution because of the special structure of the new formula.

I. INTRODUCTION

Mersenne and Fermat transforms have been developed for fast circular convolution with applications to digital signal processing using binary logic implementation without round-off error [1-12]. Using modulo a Fermat number,

$$F_t = 2^{2^t} + 1,$$

and binary logic implementation, the maximum length of the sequences which can be convolved is $2 * 2^t$ for t = 0, 1, ..., 4, and $\alpha = 2$ in number theoretic transform. In seeking for longer sequence-lengths, researchers have proposed to use the square of α being 2 in a ring of integers modulo a Fermat number [3] [13]. This special α value increases the maximum length of sequences which can be convolved by a factor of 2 under the same hardware word-length, $2^t + 1$, for the representation of all the numbers in the ring of integers modulo F_t, i.e., the maximum sequence-length is $4 * 2^t$. In binary logic, either for $\alpha = 2$ or for $\alpha = \sqrt{2}$, it is seen that the maximum length of sequences being convolved have a linear relationship with the hardware word-length. For example, using binary logic implementation of Fermat transform modulo $F_3 = 257$, 9-bit hardware word-length is needed to obtain the maximum sequence-length of 16 as α=2 and of 32 as $\sqrt{\alpha} = 2$. If we use Fermat transform modulo $F_4 = 65537$, 17-bit hardware word-length will provide the maximum sequence-length of 32 as $\alpha=2$, and of 64 as $\sqrt{\alpha} = 2$.

Mersenne transforms employ Mersenne numbers of the form

$$M_p = 2^p - 1$$

as modulo numbers, where p is a prime. One-dimensional Mersenne transforms also have hardware word-length constraint problem in binary logic because 2 is not a primitive root for the most of Mersenne primes. Another technique of using two-dimensional implementation of a one-dimensional convolution was used to extend the maximum length of sequences convolved. Using two-dimensional implementation of a one dimensional convolution [2][4], the maximum length of sequences equals twice the square of the word-length. However, as pointed out by Rader: "The overall picture of the applicability of Fermat number transforms to computation of one-dimensional convolutions is still cloudy. The number of cases in which the Fermat number transform method of filtering one-dimensional signals is clearly superior to the standard techniques is greatly limited by the word-length and sequence-length constraints" [2]. This paper is focused on the solving of the word-length and sequence-length constraints for fast convolution without round-off error by proposing the use of:

(1) generalized modulo numbers:

$$M = p^{q^t} \pm (p - 1)$$

by properly setting p, q, t such that M is a prime;

(2) the least primitive root for the generalized modulo number M as the α in the number theoretic transform to obtain the maximum sequence length $M-1$;

(3) implementations of the circular convolution with radix-$m=\alpha$.

This paper contains seven sections. Section 2 presents a review of the circular convolution property of number theoretic transform and its inverse. The transform and its inverse have circular convolution property. Section 3 provides the new generalized modulo number formula. In section 4, it is shown that the structure of the generalized modulo number formula provides simple hardware implementations of the circular convolution using radix m, and the implementation methods of arithmetic operations for fast convolution using generalized modulo number system (GMN) with $M = p^{q^t} - (p - 1)$ are proposed. The representation of n^{-1} is also given in that section. Section 5 gives an example of circular convolution. The implementations of the modulo arithmetic operations using GMN system with $M = p^{q^t} + (p - 1)$ are given in section 6. Section 7 gives a conclusion of this new approach.

II. NUMBER THEORETIC TRANSFORMS WITH CIRCULAR CONVOLUTION PROPERTY

The transform defined here transforms an n-vector of integers into another n-vector of integers within a ring of integers modulo an integer M to obtain the circular convolution property.

For $n \geq 1$, let α be a primitive nth root of unity, and let T_n be the $n \times n$ matrix with entries $t_{ij} = \alpha^{ij}$, where $0 \leq i,j \leq n-1$, $\alpha^c - 1$ is relatively prime to M, c is an integer, $1 \leq c \leq n-1$, and n is the least positive integer such that

$$\alpha^n \equiv 1 \ (mod\ M)$$

where the symbol $\equiv$ denotes the congruence. The transform of the n-vector $U = (u_0, u_1, ..., u_{n-1})$ is defined as the product $T_n U$ by modular arithmetic operations. The ith component of $T_n U$ is

$$U_i = \sum_{j=0}^{n-1} u_j \alpha^{ij} \ (mod\ M) \tag{1}$$

where $i = 0, 1, ..., n-1$.

The inverse transform T_n^{-1} is defined as an $n \times n$ matrix with entries $t_{ij}^{-1} = n^{-1}\alpha^{-ij}$, where $0 \le i, j \le n-1$, $\alpha^c - 1$ is relatively prime to M, c is an integer, $1 \le c \le n-1$, and $\alpha^n \equiv 1$. The ith component of $T_n^{-1}(T_n V)$ is

$$v_i = n^{-1} \sum_{j=0}^{n-1} V_j \alpha^{-ij} \ (mod\ M) \tag{2}$$

where $i = 0, 1, ..., n-1$.

The existence of n can be determined by number theory[20]. According to Ore, for a given modulo M there exist exponents such that

$$\alpha^{n_1} \equiv 1 \ (mod\ M) \tag{3}$$

for every α relatively prime to M. "α is relatively prime to M" means the greatest common divisor of (α, M) is 1. Euler's ϕ-function, $\phi(M)$ that gives how many numbers which are less than and relatively prime to M, is an instance of n_1. n_1 is called a universal exponent (*mod M*). Among those possible values for n_1, there is a minimal universal exponent. We denote it by n_2, and

$$\alpha^{n_2} \equiv 1 \ (mod\ M) \tag{4}$$

for every α relatively prime to M. In order to obtain a fast transform and a fast convolution algorithm in a ring of integers, we choose α to be a primitive nth root of unity (*mod M*), and both α and $\alpha^c - 1$ are relatively prime to M, for each integer c, where $1 \le c \le n-1$. Then, for such an individual α value, there exists an n, the least positive integer, such that

$$\alpha^n \equiv 1 \ (mod\ M). \tag{5}$$

If $n \ne n_2$, n must divide n_2.

It can be shown that $T_n T_n^{-1} = I$. Furthermore, the transforms and inverse transforms defined by Eqs. (1) and (2) have the circular convolution property. Namely,

$$U \otimes V = T_n^{-1}(T_n U * T_n V) \ (mod\ M) \tag{6}$$

where $\otimes$ denotes the convolution, and * denotes point-wise multiplication.

Considering the implementation with low time complexity, high accuracy, and efficient hardware utility, the radix m should be chosen as the value of α, the primitive nth root of unity. In modulo arithmetic, we have to satisfy the condition that $\alpha^c - 1$ is relatively prime to M for each integer c, $1 \le c \le n-1$. It is necessary and sufficient condition for obtaining $T_n T_n^{-1} = I$. Based on these conditions, the modulo M value is restricted. In general, modulo M does not have to be a prime number. However, if the modulo is a prime, α will not be highly restricted and all necessary conditions indicated above are easily satisfied.

III. GENERALIZED MODULO PRIMES (GMP)

The generalized modulo primes (GMP) may be generated by the following equation:

$$M = p^{q^t} \pm (p-1) \tag{7}$$

for some t such that M is a prime, where p, q are integers. Let $p = q = 2$, then $M = p^{q^t} + (p-1)$ is reduced to a Fermat number. Let $p = 2$, $t = 1$, and properly set q to be a prime such that q^t is a prime. Then $M = p^{q^t} - (p-1)$ is reduced to a Mersenne number [21]. We present the use of $M_1 = p^q - (p-1)$ first. The use of $M_2 = p^{q^t} - (p-1)$ will be discussed later. In order to obtain both simple implementation of circular convolution and the maximum sequence-length which can be convolved, we propose the use of $p = m = \alpha$ = least primitive root(LPR) for prime M. For example, the prime 7 may be generated by $M = 3^{2^1} - (3-1)$, i.e., p=3, q=2, and t=1, where 3 is also the LPR for the prime 7. By setting $\alpha = 3$, the maximum sequence-length which can be convolved in this case is 6. As another example, for $p = 3$, $q = 4$, and $t = 1$, $M = 3^{4^1} - (3-1) = 79$, the LPR for prime 79 is 3. When $\alpha = 3$ and radix-3 arithmetic are used in the implementation of the convolution, 4-bit(the term "bit" is used for any radix in this paper) hardware word-length can assure the maximum sequence-length to be $78 (mod\ 79)$.

Furthermore, consider another example, let $p = q = 3$, $t = 2$. Then, $M = p^{q^t} - (p-1) = 19681$. Note that the LPR for the prime 19681 is 11. In this case, we may but do not need to use radix-11 arithmetic; we may use a radix less than 11. For example, if radix-3 arithmetic is used for the implementation, not only the implementation remains reasonably simple, but also the length of the sequence that can be convolved remains long enough (it is equal to 9840 with a 9-bit hardware word-length) for general practical application purposes.

From the above examples, it is clearly seen that the maximum sequence-length has an exponential relationship with word-length if we properly select the radix. This is a tremendous improvement over the prior art in achieving a maximum sequence-length being convolved. A scheme of implementation of fast circular convolution proposed herein may be realized either by binary logic using coding technique [14] or by multiple-valued logic [15-19]. To-date, advanced micro-electronic techniques can provide the most basic multi-valued logic circuits for radices 3 and 4. It will be seen that if we choose $m = \alpha$, then the implementation of the convolution will have low time complexity.

The GMP representations and schemes for the implementations of the basic arithmetic operations which are required in the number theoretic transforms and the circular convolution are proposed below.

IV. MODULO ARITHMETIC OPERATIONS WITH $M = p^{q^t} - (p-1)$

In this section, the implementations of arithmetic operations for the transforms, inverse transforms, and convolution of the two sequences, are presented. Mainly, the implementations include number representation, addition, subtraction, multiplication by a power of m, general multiplication and n^{-1} in the ring of integers. Assume that the range $[0, M/2]$ is used to represent zero and positive numbers, and range $(M/2, M)$ is used to represent negative numbers. The integers whose absolute values do not exceed $M/2$ can be exactly represented in the ring of integers modulo M.

1. Number Representation

Use m's complement modulo M to represent the numbers. Then, the positive number $D = d_t d_{t-1} \cdots d_0$ in radix-m arithmetic is

$$D = \sum_{i=0}^{t} d_i m^i, \quad d_i = 0, 1, ..., m-1, \tag{8}$$

where the absolute value of D should not exceed $M/2$ as we proposed. The negative number in the ring of integers modulo M in radix-m arithmetic is

$$-D = -\sum_{i=0}^{t} d_i m^i = M - \sum_{i=0}^{t} d_i m^i \ (mod\ M). \tag{9}$$

We can implement $-D$ by setting

$$-D = M + m's\ complement\ of\ D\ (mod\ M) \tag{10}$$

and discarding the carry bit from the most significant bit (the term "bit" used here for any m value), where m's complement is defined to be similar to 2's complement in binary logic.

2. Addition/Subtraction

An addition operation in modulo M arithmetic for all three cases can be implemented by a conventional addition, and a subtraction can be performed as an addition except negating the subtrahend. If the sum exceeds modulo M, an additional addition of adding the m's complement of the modulo M to the non-modulo addition sum is necessary for residue reduction, where the carry from the most significant bit is discarded. It is clearly seen that the time complexity of an addition/subtraction operation in radix-m arithmetic is equal to that in radix-2 arithmetic.

3. Multiplication by a Power of m

Multiplication by a power of m is required frequently as a transform proceeds. As $p = m = 2$ or a power of 2, the multiplication by a power of m can be implemented by circular rotation shifts for $M = 2^{q^t} - 1$ where q^t is a prime. For any radix-m, the following implementation method is proposed.

The Shifts and Additions Method

Using radix-m arithmetic, k-bit hardware word-length is required to represent the numbers from 0 to $M-1$ in a ring of integers modulo M. k can be determined by the following inequality: $(m^k-1) \geq M > (m^{k-1}-1)$, where m and M are known. For $M = p^q - (p-1)$ with $p = m$ case, multiplication by m requires a shift of one position left and an addition of r to it, where

$$r = c\,(p-1). \tag{11}$$

for $1 \leq c \leq m-1$, and c is the carry at $(k+1)$th position. r can be precalculated under given M, m, k and the possible values of c. The final product needs to be normalized for modulo M (residue reduction). The above arithmetic operations can be implemented by hardware devices: shifter, adder, and a simple encoding circuit to generate the value of r in correspondence with the input value of c. An example is given below to show this method.

Example 1 Multiply 7209 by 3^2 *in a ring of integers modulo* $M = 19681$.

Let $M = 3^{3^2} - 2 = 19681$, and $\alpha = m = 3$. Then $k = 9$, and $r = c\,(p-1)$.

$7209 * 3^2 = 64881 = 5838\ (mod\ 19681)$

$$\begin{array}{cccr}
 & 7209 & = & 100220000 \\
 & \text{shift left} & = & \underline{1}002200000 \\
+ & \text{r} & = & \dashrightarrow 2 \\
\hline
 & & & 002200002 \\
 & \text{shift left} & = & 022000020
\end{array}$$

No addition since c=0.

$$Product = (022000020)_{m=3} = 5838\ (mod\ 19681).$$

In summary, the time complexity of a multiplication by a power of m in radix-m arithmetic is comparable to that in radix-2 arithmetic using Fermat number [5] [13]. The only difference is that the most significant bit being shifted out has $(m-1)$ possible values. But this fact does not substantially affect the time complexity herein. A simple encoding circuit is used for radix-m arithmetic instead of a negation gate in binary logic for $\alpha = m = 2$ [5].

4. General Multiplication Using $M = p^{q^t} - (p-1)$, *with* $p = m$

The product of non-modulo multiplication of two integers modulo M is stored in a radix-m register with length equal to $2q^t$ bits. We denote the high and low half register by P_h and P_l . Thus, the residue reduction of the product with modulo M may be written as

$$P = P_l + P_h * m^{q^t}$$

$$= P_l + (m-1)P_h \ (mod\ M) \qquad (12)$$

For $m = 2$, $P = P_l + P_h$ which is reduced to a Mersenne number (q^t is a prime only). If $m = 3$, $P = P_l + 2P_h$, $2P_h$ can be implemented by 2 additions. For large values of m, $(m-1)P_h$ can be implemented by $mP_h - P_h$ in which mP_h is a multiplication of power m. Example 2 is given to illustrate this method.

Example 2 Multiply 19600 by 19680 in a ring of integers modulo m = 19681.

Let $\alpha = m = 3$, $q = 3$, $t = 2$. Then $M = m^{q^t} - (m-1) = 19681$.
$19600 * 19680 = 385728000 = 81\ (mod\ 19681)$
$385728000 = (222212211000100020)_{m=3}$
$P_l = 000100020$, $P_h = 222212211$
$Product = (000010000)_{m=3} = 81\ (mod\ 19681)$.

5. The Representation of n^{-1}

According to number theory, n^{-1} exists in a ring of integers modulo M if and only if n is relatively prime to M. If modulo M is chosen as a prime and α is the primitive root for the prime, n equals Euler ϕ-function of the prime M, n^{-1} is always exists in the ring of integers modulo M, and furthermore $n^{-1} = n$.

It should be noted that n^{-1} is not always equal to n. For some prime M, e.g. 19681, note that the LPR for the prime 19681 is 11. In this case, we may but do not need to choose $\alpha = 11$; we may choose $\alpha = 3$. Then, n does not equal $\phi(M)$; n^{-1} may still exists in the ring of integers as shown in the following example.

Example 3 Find n^{-1} *for the prime* $M = 19681$ ($M = 3^{3^2} - (3-1) = 19681$) with $\alpha = 3$.

3 is of order 9840 (mod 19681), i.e., $n = 9840$. Then,
$n = 9840$, $n^{-1} = 19679\ (mod\ 19681)$
$n * n^{-1} = 193641359 = 1\ (mod\ 19681)$

We now give an example of the circular convolution.

V. AN EXAMPLE OF CIRCULAR CONVOLUTION

This is an example to compute the circular convolution using a generalized modulo number. Let $p = 3$, $q = 2$, $t = 1$, then $M = 3^{2^1} - (3-1) = 7$, 3 is of order 6, $3^6 \equiv 1$, and $3^c - 1$ is relatively prime to 7 for each c, $1 \le c \le 5$. Then,

$$T_n = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 3^2 & 3^3 & 3^4 & 3^5 \\ 1 & 3^2 & 3^4 & 3^6 & 3^8 & 3^{10} \\ 1 & 3^3 & 3^6 & 3^9 & 3^{12} & 3^{15} \\ 1 & 3^4 & 3^8 & 3^{12} & 3^{16} & 3^{20} \\ 1 & 3^5 & 3^{10} & 3^{15} & 3^{20} & 3^{25} \end{bmatrix} \pmod{7}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 6 & 4 & 5 \\ 1 & 2 & 4 & 1 & 2 & 4 \\ 1 & 6 & 1 & 6 & 1 & 6 \\ 1 & 4 & 2 & 1 & 4 & 2 \\ 1 & 5 & 4 & 6 & 2 & 3 \end{bmatrix} \pmod{7}$$

Since 3 is a primitive root of prime 7, $n^{-1} = n = \phi(7) = 6$.

$$T_n^{-1} = 6 * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3^{-1} & 3^{-2} & 3^{-3} & 3^{-4} & 3^{-5} \\ 1 & 3^{-2} & 3^{-4} & 3^{-6} & 3^{-8} & 3^{-10} \\ 1 & 3^{-3} & 3^{-6} & 3^{-9} & 3^{-12} & 3^{-15} \\ 1 & 3^{-4} & 3^{-8} & 3^{-12} & 3^{-16} & 3^{-20} \\ 1 & 3^{-5} & 3^{-10} & 3^{-15} & 3^{-20} & 3^{-25} \end{bmatrix} \pmod{7}$$

$$= 6 * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 4 & 6 & 2 & 3 \\ 1 & 4 & 2 & 1 & 4 & 2 \\ 1 & 6 & 1 & 6 & 1 & 6 \\ 1 & 2 & 4 & 1 & 2 & 4 \\ 1 & 3 & 2 & 6 & 4 & 5 \end{bmatrix} \pmod{7}$$

Let $U = (u_0, u_1, u_2, u_3, u_4, u_5) = (1, 2, 0, -1, 1, 1)$ and $V = (v_0, v_1, v_2, v_3, v_4, v_5) = (1, -1, 1, 0, 1, -2)$. Recall that we use first and second halves of the ring of integers to represent positive and negative numbers, respectively. Only numbers whose absolute values do not exceed $\frac{M-1}{2}$ can be precisely represented.

$$T_n U * T_n V = \begin{bmatrix} 0 \\ 3 \\ -9 \\ 0 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 5 \\ 0 \\ 0 \\ 2 \end{bmatrix} \pmod{7}$$

$$T_n^{-1}(T_n U * T_n V) = 6 * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 4 & 6 & 2 & 3 \\ 1 & 4 & 2 & 1 & 4 & 2 \\ 1 & 6 & 1 & 6 & 1 & 6 \\ 1 & 2 & 4 & 1 & 2 & 4 \\ 1 & 3 & 2 & 6 & 4 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 5 \\ 0 \\ 0 \\ 2 \end{bmatrix} \pmod{7}$$

$$= 6 * \begin{bmatrix}10\\41\\26\\35\\34\\29\end{bmatrix} = \begin{bmatrix}4\\1\\2\\0\\1\\6\end{bmatrix} = \begin{bmatrix}-3\\1\\2\\0\\1\\-1\end{bmatrix} \quad (mod\ 7)$$

From the above example, it is shown that the use of $\alpha = m = p = LPR$ extends the maximum sequence-length of two sequences being convolved to $(M - 1)$ in a ring of integers modulo the prime M. If Mersenne number $M = 2^3 - 1 = 7$, and $\alpha = m = 2$ are used in this case, the maximum sequence-length of two sequences which can be convolved is reduced to 3, the half of 6, since $2^3 \equiv 1 \pmod 7$. Moreover, the use of radix-3 arithmetic requires only 2-bit hardware word-length instead of 3-bit hardware word-length in radix-2 arithmetic.

VI. MODULO ARITHMETIC OPERATIONS WITH $M = p^{q'} + (p-1)$

The use of $M = p^{q'} + (p-1)$ is similar to the use of $M = p^{q'} - (p-1)$. The best choice for us is still $p = \alpha = m$ = the least primitive root for the prime M. The implementations of arithmetic operations are the same as those for $M = p^{q'} - (p-1)$ except the implementation methods of multiplications. The differences are given below.

1. Multiplication by a power of m

As we mentioned in the previous section, using radix-m arithmetic, k-bit hardware word-length is required to represent the numbers from 0 to $M-1$ in a ring of integers modulo M. k can be determined by the following inequality: $(m^k-1) \geq M > (m^{k-1}-1)$, where m and M are known. Multiplication by m requires a shift of one position left and an addition of r to it, where

$$r = cm^k - \left\lfloor \frac{cm^k}{M} \right\rfloor *M \tag{13}$$

for $1 \leq c \leq m-1$, and c is the carry at $(k+1)$th position. r can be precalculated under given M, m, k and the possible values of

2. General Multiplication

(1) $M = p^{q'} + (p-1)$, with $p = m$

The product of non-modulo multiplication of two integers modulo M is stored in an m-valued register with length equal to $2q' + 1$ bits. Then we denote the low q' bits by P_l, and the rest bits by P_h. The residue reduction of the product with modulo M may proceed as

$$\begin{aligned} P &= P_l + P_h * m^{q'} \\ &= P_l - (m-1)P_h \end{aligned} \tag{14}$$

(2) $M = p^{q'} + (p - 1)$ with $m \neq p$

Without loss of generality, suppose $p = q = 2$. M is a Fermat number. For m ≠ 2, the implementation is not as straightforward as for $p = m$. The authors proposes a procedure described as follows:

(i) Store the product of non-modulo multiplication of two integers both modulo M in an m-valued register with length of $2k + 1$ bits, where k is determined by

$$m^k - 1 \geq M > m^{k-1} - 1 \qquad (15)$$

(ii) Let $R = m^k - Mk_1$, where k_1 is determined by

$$M(k_1 + 1) > m^k \geq Mk_1. \qquad (16)$$

Then, the general multiplication of modulo arithmetic in radix-m arithmetic can be completed by

$$P = P_l + c_0 R + c_1 mR + c_2 m^2 R + \cdots + c_k m^k R \ (mod\ M) \qquad (17)$$

with final residue reduction, where $c_0, c_1, ..., c_k$ are the bits at bit weight $m^{k+0}, m^{k+1}, ..., m^{2k}$, respectively. It is clear that once M and m are given, k, k_1, and R can be precalculated. Furthermore, $c_i m^i R$, $i \in \{0, 1, ..., k\}$ can be precalculated for any c_i, $c_i \in \{0, 1, ..., m\text{-}1\}$. An example is given below. Obviously, A similar method can be derived for $M = p^q - (p-1), m \neq p$.

VII. CONCLUSION

This paper has offered a new solution to the sequence-length constraint problem in one-dimensional convolution using number theoretic transform. The method provides not only more choices of modulo numbers for number theoretic transform, but also simple implementation of fast circular convolution because of the special structure of the new formula. The implementation methods of modulo arithmetic operations: number representation, addition/subtraction, multiplication by a power of m, and general multiplication, have been proposed for calculating number theoretic transform and circular convolution.

As with the Mersenne and Fermat transforms, the circular convolution obtained by the new approach is perfectly accurate, i.e., without round-off error.

ACKNOWLEDGMENT

The authors wish to thank C. M. Rader of Lincoln Laboratories at Massachusetts Institute of Technology, Lexington, Massachusetts, for his valuable comments and suggestion.

REFERENCES

[1] Rader, C. M., "Discrete Convolution via Mersenne Transforms," *IEEE Trans. Comput.*, vol. c-21, No. 12, 1972, pp. 1269-1273.

[2] Rader, C. M., "On the Application of the Number Theoretic Transforms of High Speed Convolution to Two-Dimensional Filtering," *IEEE Trans. Circuits and Systems,* vol. CAS-22, June 1975, p. 575.

[3] Agarwal, R. C. & Burrus, C. S., "Fast Convolution Using Fermat Number Transforms with Applications to Digital Filtering," *IEEE Trans. Acoust., Speech, and Signal Processing,* vol. ASSP-22, No. 2, 1974, pp. 87-97.

[4] Agarwal, R. C. & Burrus, C. S., "Fast One-Dimensional Digital Convolution by Multidimensional Techniques," *IEEE Trans. Acoust., Speech, and Signal Processing,* vol. ASSP-22, No. 1, 1974, pp. 1-10.

[5] Leibowitz, L. M., "A Simplified Binary Arithmetic for the Fermat Number Transform," *IEEE Trans. Acoust., Speech, and Signal Processing,* vol. ASSP-24, No. 5, 1976, pp. 356-359.

[6] Chevillat, P. R., "Transform-Domain Digital Filtering With Number Theoretic Transforms and Limited Word Lengths," *IEEE Trans. Acoust., Speech, Signal*

Processing, vol. ASSP-26, 1978, pp. 284-290.

[7] Nussbaumer, H. J., "Digital Filtering Using Pseudo Fermat Number Transform," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-25, 1977, pp. 79-83.

[8] Brule, J. D., "Fast Convolution with Finite Field Fast Transforms," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-23, 1975, p. 240.

[9] Reed, I. S. & Truong, T. K., "The Use of Finite Fields to Compute Convolutions," *IEEE Trans. Inform. Theory,* vol. IT-21, 1975, pp. 208-213.

[10] Nussbaumer, H. J., "Digital Filtering Using Complex Mersenne Transforms," *IBM J. Res. Dev.,* vol. 20, 1976, pp. 498-504.

[11] Nussbaumer, H. J., *Fast Fourier Transform and Convolution Algorithms,* Springer-Verlag. Berlin(1981).

[12] Morhac, M., "Precise Deconvolution Using the Fermat Number Transform," *Comput. & Maths. with Appls.,* vol. 12A, No. 3, 1986, pp. 319-329.

[13] McClellan, J. H., "Hardware Realization of a Fermat Number Transform," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-24, No. 3, 1976, pp. 216-225.

[14] McCluskey, E. J., "A Discussion of Multiple-Valued Logic Circuits," *IEEE Proc. 12th International Symposium on Multiple-Valued Logic,* 1982, pp. 200-205.

[15] Allen, C. M. & Givone D. D., "The Allen-Givone Implementation Oriented Algebra," [D. C. Rine Ed., 2nd Ed.] *Computer Science and Multiple-valued Logic Theory and Applications,* North-Holland, 1984, pp. 268-281.

[16] Su, S. & Cheung, P. T., "Computer Simplification of Multi-valued Switching Functions," [D. C. Rine Ed., 2nd Ed.] *Computer Science and Multiple-valued Logic Theory and Applications,* North-Holland, 1984, pp. 195-226.

[17] Vranesic, Z. G. & Lee, E. S. & Smith, K. C., "A Many-valued Algebra for Switching Systems," *IEEE Trans. Comput.,* vol. c-19, 1970, pp. 964-971.

[18] Rine, D. C., *Computer Science and Multiple-Valued Logic, Theory and Applications,* North-Holland, 1984.

[19] Lu, H. & Lee, S. C., "M - algebra," *IEEE Proc. 15th International Symposium on Multiple-Valued Logic,* 1985, pp. 272-285.

[20] Ore, O., *Number Theory and Its History,* New York:McGraw Hill, 1948.

[21] Lu, H. & Lee, S. C., "Fast Convolution Using Generalized Fermat/Mersenne Number Transforms," *IEEE Proc. International Conference on Acoustics, speech, and Signal Processing,* 1988, pp. 1910-1913.

A FAMILY OF OPTIMAL ARCHITECTURES FOR MULTIDIMENSIONAL TRANSFORMS

CHAITALI CHAKRABARTI and JOSEPH JÁJÁ
Department of Electrical Engineering
Systems Research Center
University of Maryland
College Park, MD 20742.

ABSTRACT

Multidimensional transforms have widespread applications in computer vision, pattern analysis and image processing. The only existing optimal architecture for computing multidimensional DFT on data of sixe $n = N^d$ requires very large rotator units of area $O(n^2)$ and pipeline-time $O(\log n)$. In this paper we propose a family of optimal architectures with area-time trade-offs for computing multidimensional transforms. The large rotator unit is replaced by a combination of a small rotator unit, a transpose unit and a block rotator unit. The combination has an area of $O(N^{d+2a})$ and a computation time of $O(N^{\frac{d}{2}-a}\log n)$, for $0 < a \leq \frac{d}{2}$. A design with a low value of a has less area, whereas a design with a large value of a requires less computation time.

1 INTRODUCTION

Multidimensional transforms are a powerful tool for analyzing multidimensional signals. The 2-D Discrete Fourier Transform (DFT) is widely used in spectrum analysis, speech processing and image processing. The 3-D and 4-D DFTs are used to represent dynamic patterns in computer vision and pattern analysis. Since the number of computations involved in such transforms is very large, optimal architectures with efficient computational schemes are needed.

There exists an architecture [4] for computing multidimensional DFT on a single file of $n = N^d$ elements whose AT_p^2 performance achieves the known lower bound, where A is the area and T_p is the pipeline time. The design consists of N^{d-1} DFT(N) computation units and a rotator of size $O(n^2)$ for data permutation. In this paper [2] we show that if the input is in the form of a 2-D array of size $N^{\frac{d}{2}+a} \times N^{\frac{d}{2}-a}$, $0 < a \leq \frac{d}{2}$, then we can design a *family of optimal architectures* for different values of a. The design of [4] is a member of this family when $a = \frac{d}{2}$. There are two cases depending on whether A) a is an integer or B) a is not necessarily an integer but N^a is an integer. Our design for Case A consists of $N^{\frac{d}{2}+a-1}$ DFT(N) arrays, a transpose unit of size $O(N^d \log n)$ and a rotator of size $O(N^{d+2a})$. Case B, which is slightly more complicated, requires an additional subblock rotator unit of size $O(N^{d+2a})$. The maximum computation time, T, for both the cases is $O(N^{\frac{d}{2}-a}\log n)$. Thus for low values of a the area would be small, whereas for large values of a the computation time would be small. In addition all these designs satisfy the AT^2 lower bound.

The rest of the paper is organized as follows. In Section 2 we state the definitions and lower bounds for the computation of multidimensional linear transforms. Existing architectures are briefly discussed in Section 3. Section 4 deals with the family of optimal architectures for different values of a. Section 5 summarizes the whole paper.

2 PRELIMINARIES

Let n be the total number of data elements that are to be organized in a d-dimensional data cube. Each element has to be represented by d indexes $n_1, n_2, \ldots, n_d$. Any d-dimensional separable linear transform can be defined by

$$X(k_1, k_2, \ldots, k_d) = \sum_{n_d} \cdots \sum_{n_2} \sum_{n_1} x(n_1, n_2, \ldots n_d)\alpha_1(n_1, k_1)\alpha_2(n_2, k_2)\ldots\alpha_d(n_d, k_d)$$

where the α_is are the transform functions, $0 \le k_i, n_i \le N_i - 1$ for $1 \le i \le d$ and $n = N_1 N_2 \ldots N_d$. In order to simplify our analysis, we assume

1. $N_1 = N_2 = \cdots = N_d = N$ and $n = N^d$
2. N is a power of 2, that is, $N = 2^m$.

We proceed to state along the lines of [4] the lower bound on AT^2 for multidimensional DFT, DHT and DCT. The computation time, T, is used to describe the time performance of any circuit. We assume that if n be the problem size, then O(log$n$) bits are sufficient to represent the value of a variable. Vuillemin [6] has shown that $AT^2 = \Omega(I^2)$ for any chip computing a transitive problem of size I. For multidimensional DFT, DHT and DCT, the information content of a problem I is $\Omega(n \log n)$. Thus the lower bound of AT^2 is $\Omega(n^2 \log^2 n)$.

3 EXISTING ARCHITECTURES

In this section we briefly discuss the various architectures for computing multidimensional linear transforms. There exist schemes for computing 1-D as well as multidimensional DFT in the literature. However, to the authors' knowledge, there are no known schemes for computing multidimensional DHT and DCT.

The schemes for computing 2-D DFT are based on computing 1-D DFT on columns followed by 1-D DFT computation on rows. These schemes require either a separate array transpose unit [3], [5] or an internal scheme for transposing the data [7]. All these designs have an AT^2 of $O(n^2 \log^3 n)$, which is $O(\log n)$ away from the optimal. Gertner and Shamash [4] have recently proposed an optimal architecture for multidimensional Fourier transform. Their design consists of N^{d-1} butterfly arrays for computing 1-D DFT(N) and a rotation network array or rotator for permuting the data. The rotator is based on a network which performs rotation over an index in one step. The data enter the rotator in n rows. It is then rotated with the help of load-shift cells placed at the connection of a row input and its column necklace. The number of columns required to layout the rotator is O(n). The area complexity is O(n^2) and the pipeline time complexity is O($\log n$). Thus this scheme achieves the AT_p^2 lower bound. The computation time T is $O(d\log^2 n)$.

The large size of the rotator unit and the large number of 1-D DFT(N) circuits makes the design of [4] unattractive. For moderately large n, the size of the interconnection would be huge. This would also mean complicated inter-chip and intra-chip connections. Moreover, since $T = O(d\log^2 n)$, the bounds for AT^2 are not optimal. We can reduce the area of the design drastically at the expense of a larger computation time and still achieve the AT^2 lower bound. In the next section we will show that there exists a family of optimal architectures, of which the design of [4] is a member.

4 FAMILY OF OPTIMAL ARCHITECTURES

In the design by Gertner and Shamash [4], a single file of $n = N^d$ elements are processed by N^{d-1}DFT(N) circuits d times. The algorithm for d-dimensional DFT is summarized as follows.

1. Repeat steps 2 and 3 d times
2. Compute DFT(N) for N^{d-1} vectors
3. Rotate over an index

In our design the data is organized in a 2-D array with $N^{\frac{d}{2}+a}$ rows and $N^{\frac{d}{2}-a}$ columns. The bounds of a are $0 < a \leq \frac{d}{2}$. Without loss of generality, we assume that d is divisible by 2. When $a = \frac{d}{2}$, the 2-D array collapses into a single column of N^d elements. This is the case investigated by [4]. When d is not divisible by 2, the data can be organized into a 2-D array with $N^{\lceil d/2 \rceil + a}$ rows and $N^{\lfloor d/2 \rfloor - a}$ columns. The advantage of having a 2-D data array is that the same DFT computation unit can be used to compute DFT of multiple columns. We shall investigate two cases here :

A. When a is an integer

B. When a is not necessarily an integer but N^a is an integer.

We develop the architectures for Cases A and B in Sections 4.1 and 4.2 respectively.

4.1 Case A : a is an integer

The data is organized in column major order in a 2-D array of size $N^{\frac{d}{2}+a} \times N^{\frac{d}{2}-a}$. In this representation all the elements in a column will have the same value for indexes $n_{\frac{d}{2}+a+1}, \ldots, n_d$ and all the elements in a row will have the same value for indexes $n_1, \ldots, n_{\frac{d}{2}+a}$. The d-dimensional DFT(N) is computed by evaluating the $(\frac{d}{2}+a)$-dimensional DFT along the columns followed by evaluating the $(\frac{d}{2}-a)$-dimensional DFT along the rows. The algorithm is as follows.

1. For each column perform $(\frac{d}{2}+a)$-dimensional DFT
2. Transpose
3. For each column perform $(\frac{d}{2}-a)$-dimensional DFT

In the first step DFT(N) is computed over indexes $n_1, n_2, \ldots, n_{\frac{d}{2}+a}$. This is possible by circulating the data $(\frac{d}{2}+a)$ times through the DFT(N) computation unit and the rotator unit ROT1. The theory and layout of ROT1 are very similar to that of the rotator of [4].

The direction of rotation is such that index n_i becomes index $n_{(i-2)mod(\frac{d}{2}+a)+1}$, $1 \leq i \leq \frac{d}{2}$. This guarantees that the same rotator can be used to compute $(\frac{d}{2}+a)$ as well as $(\frac{d}{2}-a)$-dimensional DFT. Since the number of input lines to the rotator is $N^{\frac{d}{2}+a}$, the size of the rotator is $O(N^{d+2a})$. The number of DFT(N) circuits required is $N^{\frac{d}{2}+a-1}$ compared to N^{d-1} of [4].

In the next step the 2-D data array of size $N^{\frac{d}{2}+a} \times N^{\frac{d}{2}-a}$ is transposed. A simple transpose unit [3],[5] would have an area of $O(n \log n)$ and computation time of $O(N^{\frac{d}{2}+a} \log n)$. The design would then have an AT^2 which is $O(N^{4a})$ away from the optimal. In our design the transpose unit is constructed so that the computation time is reduced to $O(N^{\frac{d}{2}-a} \log n)$. This is done in the following way. The data block is partitioned into N^{2a} subblocks, each of size $N^{\frac{d}{2}-a} \times N^{\frac{d}{2}-a}$. Since the subblock has equal number of elements in rows and columns, by transposing N^{2a} subblocks parallely, the size of the transposed data block does not change. The transpose unit in our design consists of N^{2a} subunits, each of which operates on a data subblock. Since the subunits operate in parallel, T for the transpose unit is $O(N^{\frac{d}{2}-a} \log n)$. The total area of the transpose unit is $N^{2a}O(N^{d-2a} \log n) = O(n \log n)$. After transpose all the elements in a column of a subblock have the same value for indexes $n_1, \ldots, n_{\frac{d}{2}+a}$ but different values for indexes $n_{\frac{d}{2}+a+1}, \ldots, n_d$.

In the third step DFT(N) is computed over indexes $n_{\frac{d}{2}+a+1} \ldots, n_d$ by circulating the data $(\frac{d}{2}-a)$ times through the DFT(N) computation unit and ROT1. The block diagram for our design for this case is shown in Fig.1.

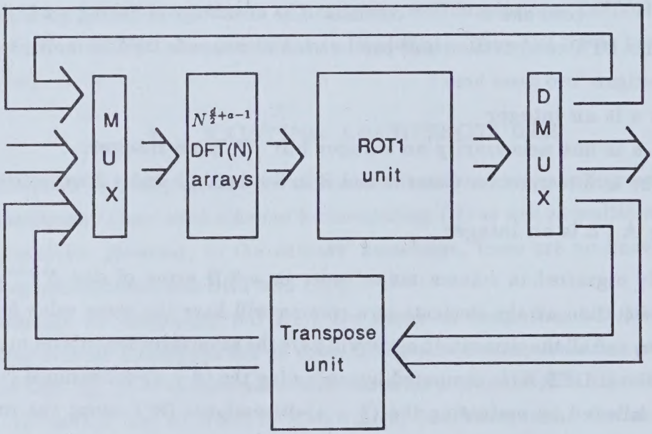


Fig.1 Architecture for Case A

The area complexity of the input MUX and the output DMUX are $O(N^{\frac{d}{2}+a})$. If we use the scheme of [1] in the computation of DFT(N), the area of the DFT(N) computation unit would be $N^{\frac{d}{2}+a-1} \times O(N^2) = O(N^{\frac{d}{2}+a+1})$. The area of ROT1 is $O(N^{d+2a})$ and that of the transpose unit is $O(N^d \log n)$. For $a \geq \frac{1}{2}\log_N(\log n)$, $N^d \log n \leq N^{d+2a}$. Thus the maximum area, $A_{max} = O(N^{d+2a})$, is contributed by the rotator unit.

The time taken to compute DFT(N) as well as to rotate a column of data once is $O(\log n)$. Thus Steps 1 and 3 require computation times of $O((\frac{d}{2}+a)N^{\frac{d}{2}-a}\log n)$ and $O((\frac{d}{2}-a)N^{\frac{d}{2}-a}\log n)$ respectively. The time taken to transpose the data in Step 2 is $O(N^{\frac{d}{2}-a}\log n)$. Thus the total computation time is $O(dN^{\frac{d}{2}-a}\log n)$ which can be approximated to $O(N^{\frac{d}{2}-a}\log n)$, for constant d. AT^2 for our design is $O(N^{2d}\log^2 n) = O(n^2\log^2 n)$. Since for all practical cases $\frac{1}{2}\log_N(\log n) \leq 1$, we can design d-dimensional DFTs optimally in the range $1 \leq a \leq \frac{d}{2}$. The number of members of this family of optimal architectures is $\frac{d}{2}$. In the special case when $a = \frac{d}{2}$, the computation time of our scheme is $O(\log n)$ less than that of [4].

4.2 Case B : a is not necessarily an integer but N^a is an integer

This case is slightly more complicated than Case A. This is because when a is not an integer, we cannot compute $(\frac{d}{2}+a)$ as well as $(\frac{d}{2}-a)$-dimensional DFT. Our scheme for Case A has to be modified appropriately.

Let $a = i + f$, where i is an integer, f is a simple fraction and N^f is an integer. We can think of the data to be organized in column major order in a 2-D array of size $N^{\frac{d}{2}+i} \times N^{\frac{d}{2}-i}$, $0 \leq i \leq \frac{d}{2}$ as shown in Fig.2a. Thus all the elements in a column have the same value for indexes $n_{\frac{d}{2}+i+1}, \ldots, n_d$, and all the rows have the same value for indexes $n_1, \ldots, n_{\frac{d}{2}+i}$. The array is partitioned vertically into N^f blocks each of size $N^{\frac{d}{2}+i} \times N^{\frac{d}{2}-a}$ and placed one after the other. Each of the blocks is further partitioned into N^{a+i} subblocks of size $N^{\frac{d}{2}-a} \times N^{\frac{d}{2}-a}$ and numbered in column major order as shown in Fig.2b. All the elements in a row have the same lower $m(\frac{d}{2}+a)$ bits and all the elements in a column have the same higher $m(\frac{d}{2}-a)$ bits. This is possible because the m bits which were used to represent the values for n_d have been split so that the least significant $m(1-f)$ bits of n_d combine with the $m(\frac{d}{2}-i-1)$ bits of indexes $n_{\frac{d}{2}+i+1}, \ldots, n_{d-1}$ and the most significant mf bits of n_d combine with the $m(\frac{d}{2}+i)$ bits of indexes $n_1, \ldots, n_{\frac{d}{2}+i}$. Since every row in Fig.2a has been split into N^f parts, there are N^f rows with the same values for indexes $n_1, \ldots, n_{\frac{d}{2}+i}$ in Fig.2b. Since $a > i$, if we use the method discussed in Case A to compute DFT(N) $\frac{d}{2}+i$ times over each column of Fig.2b followed by $\frac{d}{2}-i-1$ times over each row of Fig.2b, then DFT(N) would be computed over $d-1$ instead of d dimensions. The method of Case A has to be thus modified.

In this scheme $(\frac{d}{2}+i)$-dimensional DFT is performed on each column of a block. This means that in every block DFT(N) is computed over indexes $n_1, \ldots, n_{\frac{d}{2}+i}$ by circulating the data $\frac{d}{2}+i$ times through a DFT(N) computation unit and a ROT1 unit. There are N^f DFT(N) computation units, each consisting of $N^{\frac{d}{2}+a-1}$ DFT(N) circuits. The rotator unit consists of N^f ROT1 units each of size $O(N^{d+2i})$.

The scheme for the transpose operation is similar to that of Case A. The transpose unit consists of N^{2a} subunits each of which operates on a data subblock. The area of the transpose unit is $N^{2a}O(N^{d-2a}\log n) = O(N^d\log n)$. Since the subunits operate in parallel, the computation time is $O(N^{\frac{d}{2}-a}\log n)$.

The next operation is to compute DFT(N) over the remaining $\frac{d}{2} - i$ indexes. After the transpose operation all the elements with the same value for indexes $n_1, \ldots, n_{\frac{d}{2}+i}$ are now in the same column. If two elements were adjacent to each other in a row (that is row-adjacent) in Fig.2a then a) before transpose they were either row-adjacent in Fig.2b or separated by $N^{\frac{d}{2}+i}$ elements b) after transpose they are either column-adjacent or separated by $N^{\frac{d}{2}+i} - N^{\frac{d}{2}-a}$ elements.

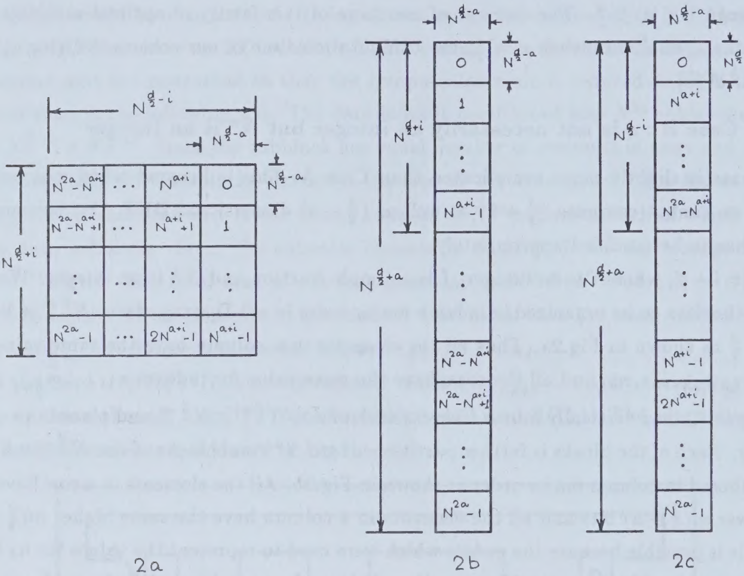

$N^{\frac{d}{2}-i}$
$N^{\frac{d}{2}-a}$
$N^{\frac{d}{2}-a}$
$N^{2a}-N^{a+i}$
N^{a+i}
0
$N^{2a}-N^{a+i}+1$
$N^{a+i}+1$
1
$N^{\frac{d}{2}+i}$
$N^{2a}-1$
$2N^{a+i}-1$
$N^{a+i}-1$
2a
$N^{\frac{d}{2}-a}$
$N^{\frac{d}{2}-a}$
0
1
$N^{\frac{d}{2}+i}$
$N^{a+i}-1$
$N^{\frac{d}{2}+a}$
$N^{2a}-N^{a+i}$
$N^{2a}-N^{a+i}+1$
$N^{2a}-1$
2b
$N^{\frac{d}{2}-a}$
$N^{\frac{d}{2}-a}$
0
N^{a+i}
$N^{\frac{d}{2}-i}$
$N^{2a}-N^{a+i}$
$N^{\frac{d}{2}+a}$
$N^{a+i}-1$
$2N^{a+i}-1$
$N^{2a}-1$
2c

Fig.2a Partitioning data of size $n = N^d$ into N^{2a} subblocks
2b Subblock ordering
2c Subblock ordering after passing through BROT1

The function of the subblock rotator unit BROT1 is to rotate subblocks of data so that all the elements which were row-adjacent in Fig.2a are column-adjacent in Fig.2c. Fig.2c illustrates the ordering of the subblocks after passing through BROT1. The data subblock is rotated in a way so that the relative position between the elements in a subblock do not change. This means that all the elements of a data subblock are displaced by the same amount. The layout of BROT1 is as follows. Since there are $N^{\frac{d}{2}-a}$ elements in every subblock column, every subblock necklace consists of $N^{\frac{d}{2}-a}$ necklaces. The total number of columns required to rotate the subblocks is $N^{\frac{d}{2}-a}O(N^{2a}) = O(N^{\frac{d}{2}+a})$. The area of BROT1 is thus $O(N^{d+2a})$. Fig.3 illustrates the layout of BROT1 for an example with 16 subblocks and 4 elements/subblock.

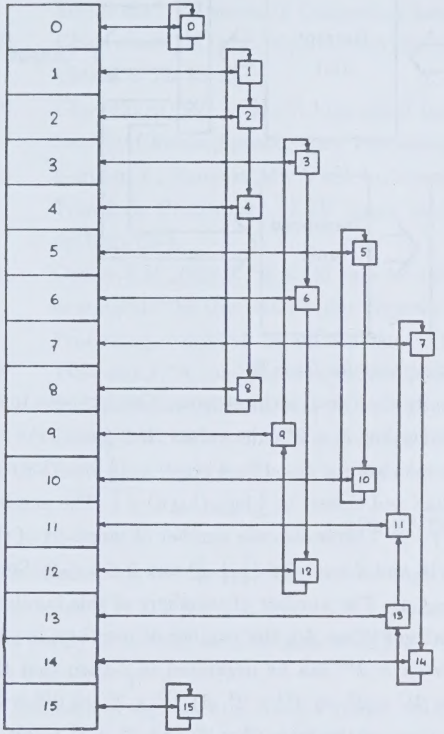
The algorithm for Case B is as follows.

1. For each column perform $(\frac{d}{2} + i)$-dimensional DFT
2. Transpose subblocks

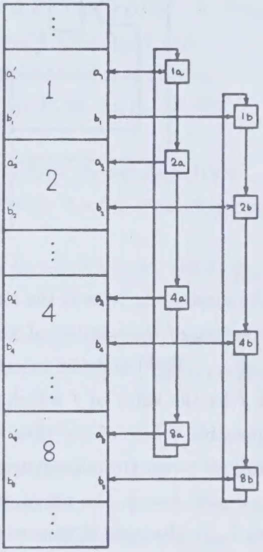
3. Rotate subblocks

4. For each column perform $(\frac{d}{2} - i)$-dimensional DFT

The block diagram for our design is shown in Fig.4. Appendix 1 illustrates the various steps for the computation of 2-D DFT(4).



3a



3b

Fig.3a Layout of 16 subblock BROT1

3b Layout of the subblock necklace 1-2-4-8-1

The computation time for the DFT(N) unit [1], the rotator unit as well as the BROT1 unit is $O(\log n)$. Thus the time taken to compute Steps 1 and 4 are $O((\frac{d}{2} + i)N^{\frac{d}{2}-a} \log n)$ and $O((\frac{d}{2} - i)N^{\frac{d}{2}-a} \log n)$ respectively. Steps 2 and 3 require a computation time of $O(N^{\frac{d}{2}-a} \log n)$. Thus the computation time for this case is $O(dN^{\frac{d}{2}-a} \log n)$, which can be approximated to $O(N^{\frac{d}{2}-a} \log n)$ for constant d.

The area of the DFT(N) unit, the input MUX and the output DMUX are the same as that of Case A. The area of the rotation unit is $O(N^{d+a+i})$. Since $a \geq i$, the BROT1 unit with an area of $O(N^{d+2a})$ dominates the area of the design. Thus $A_{max} = O(N^{d+2a})$ for $\frac{1}{2}\log_N(\log n) \leq a \leq \frac{d}{2}$. The value of $AT^2 = O(n^2 \log^2 n)$ for $\frac{1}{2}\log_N(\log n) \leq a \leq d/2$ and

N^a is an integer.

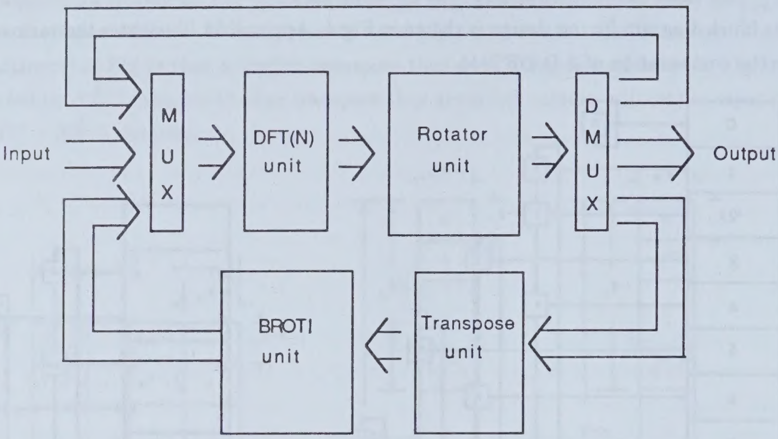


Fig.4 Architecture for Case B

An interesting issue is the size of this family of optimal architectures. This happens to be a function of the number of values a can take. For $N = 2^m$, the values that f can take are $[\frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}]$ and the values that i can take are $\hat{\imath} < i \leq d/2$, where $\hat{\imath} = \lfloor \frac{1}{2} \log_N(\log n) \rfloor$. Let $\hat{f}$ be the value of f which is greater than and closest to $\frac{1}{2}\log_N(\log n) - \hat{\imath}$. The number of possible values of a is then $m(\frac{d}{2} - \hat{\imath} - \hat{f}) + 1$. This is also the number of members of the family of optimal architectures. For $N = 16$ and $d = 4$, $f \in [\frac{1}{4}, \frac{1}{2}, \frac{3}{4}]$ and $0 \leq i \leq 2$. Since $\hat{\imath} = 0$ and $\hat{f} = \frac{1}{2}$, the range of a is $\frac{1}{2} \leq a \leq 2$. The number of members of this family is thus 7. In the special case when a is an integer (Case A), the number of members is only 2. The possible ways in which data of size $n = 2^{16}$ can be organized in 2-D so that the architecture is optimal are a) $2^{10} \times 2^6$, b) $2^{11} \times 2^5$, c) $2^{12} \times 2^4$, d) $2^{13} \times 2^3$, e) $2^{14} \times 2^2$, f) $2^{15} \times 2$ and g) $2^{16} \times 1$. Data organizations of the form $2^8 \times 2^8$ and $2^9 \times 2^7$ result in non optimal architectures, since the size of the transpose unit is greater than that of the subblock rotator unit.

5 CONCLUSION

In this paper we have proposed a family of optimal architectures with area-time trade-offs for computing multidimensional linear transforms. The only existing optimal architecture for computing multidimensional DFT on data of size $n = N^d$, requires very large rotator units of area $O(n^2)$ and pipeline time $O(\log n)$. In our design the large rotator unit is replaced by a combination of a small rotator unit, a transpose unit and a subblock rotator unit with area $O(N^{d+2a})$ and computation time $O(N^{\frac{d}{2}-a}\log n)$ for $\frac{1}{2}\log_N(\log n) \leq a \leq \frac{d}{2}$. A design with a low value of a has less area, whereas a design with a large value of a requires less computation time. We have discussed in detail the computation of multidimensional DFT. The architectures for multidimensional DHT and DCT differ only in the 1-D computation block.

REFERENCES

[1] Bilardi,G.,Sarrafzadeh,M.,"Optimal VLSI circuits for the Discrete Fourier Transform", Advances in Computing Research, vol.4, pp.87-101.

[2] Chakrabarti,C.,JáJá.,"Optimal Architectures for Multidimensional Transforms", UMIACS-TR-88-36.

[3] Chowdhury, N.U.,et.al.,"A high speed two dimensional FFT processor", in Proc. Int. Conf.Acous.,Speech,Signal Processing, San Diego,CA,1984, pp.4.11.1-4.11.4.

[4] Gertner, I., Shamash,M.,"VLSI Architectures for Multidimensional Fourier Transform Processing", IEEE Trans. on Computers, vol.C-36, no.11, Nov'87, pp.1265-1274.

[5] Owens,R.M.,JáJá,J.,"A VLSI chip for the Winograd/Prime Factor Algorithm to compute the Discrete Fourier Transform", IEEE Trans. Acoust.,Speech,Signal Processing, vol.ASSP-34, pp.979-989, Aug.'86.

[6] Vuillemin,J."A combinatorial limit to the computing power of VLSI", IEEE Trans. on Computer, vol.C-32, Mar'83, pp.294-300.

[7] Zhang,C.H.,"Multi-dimensional systolic networks for discrete Fourier transform", Proc. 11th Annu.Int.Symp.Computer Architecture, Ann Arbor, MI, 1984, pp.21-27.

APPENDIX 1

In this section we illustrate the various steps for the computation of 2-D DFT(4). Since $N = 4$ and $d = 2$, $0 \leq i \leq 1$, $0 < a \leq 1$, $\hat{\imath} = 0$ and $\hat{f} = \frac{1}{2}$. Thus the number of members of this family of optimal architectures is $m(\frac{d}{2} - \hat{\imath} - \hat{f}) + 1 = 2$. We will study the case when $i = 0$ and $f = \frac{1}{2}$.

The data is organized in column major order into a 2-D array of size 4×4 as shown in Fig.5a. Computation of 2-D DFT(4) is equivalent to computation of 1-D DFT(4) on the columns of Fig.5a followed by 1-D DFT(4) on the rows of Fig.5a. The data array is partitioned vertically into $N^f = 2$ data blocks and stacked as shown in Fig.5b. Two DFT(4) circuits are required to operate parallely on the columns of the two blocks of the data array. In this way 1-D DFT(4) is computed on the columns of Fig.5a. In order that the size of the data array remains the same before and after the transpose, the array is partitioned horizontally into N^{2a} data subblocks each of size $N^{1-a} \times N^{1-a}$, that is, 2×2. The transpose unit consista of $N^{2a} = 4$ subunits which operate on the subblocks parallely. The resulting data array is shown in Fig.5d. Note that all the elements which were in the same row in Fig.5a are now in the same column of Fig.5d though they are not adjacent. The block rotator unit rotates the data subblocks so that all the elements which were adjacent along a row in Fig.5a are now adjacent along a column of Fig.5e. 1-D DFT(4) is computed along the rows of Fig.5a by passing the data through the DFT(4) computation unit as shown in Fig.5e.

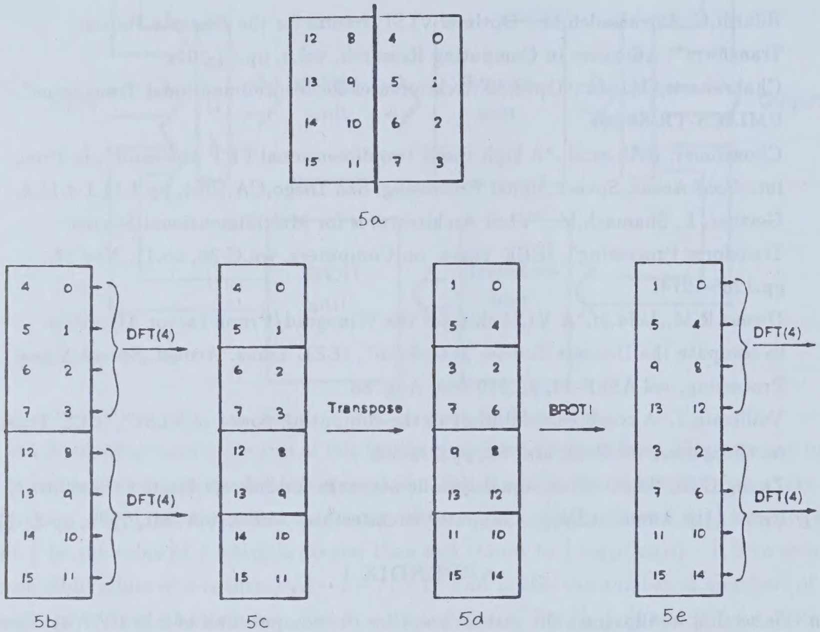

12 8 4 0
13 9 5 1
14 10 6 2
15 11 7 3
5a
4 0
5 1
6 2
7 3
12 8
13 9
14 10
15 11
DFT(4)
DFT(4)
5b
4 0
5 1
6 2
7 3
12 8
13 9
14 10
15 11
5c
Transpose
1 0
5 4
3 2
7 6
9 8
13 12
11 10
15 14
5d
BROT1
1 0
5 4
9 8
13 12
3 2
7 6
11 10
15 14
DFT(4)
DFT(4)
5e

Fig.5a Column order data organization
5b Input data organization
5c Data after computing DFT(4)
5d Data after computing transpose
5e Data after rotating subblocks

PARALLEL ARCHITECTURE FOR BOUNDARY FOLLOWING OF REGIONS OF AN IMAGE STORED AS LINEAR QUADTREE

R.B.Panwar* and S.K.Nandy**

* Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore
** Laboratory for Computer Aided Design
Indian Institute of Science, Bangalore

ABSTRACT

In this paper we present a parallel architecture for following boundary of regions represented as linear quadtrees. The architecture consists of processors arranged as a quadtree of height h, $0<h\leq(p-1)$, where the processors at level h-1 are also connected as two dimensional mesh. The processors at the level h-1 have capability to perform BFA on subimages allotted to them whereas the remaining processors have very restricted capabilities and are called as routers. The sequential BFA presented in [2] has been modified to make it more amenable to parallel implementation on the architecture. To perform BFA on an image of size n x n the parallel architecture requires O(logn) time using $O(n^2/\log^2 n)$ processors. An efficient layout of the architecture is presented which requires O(N) area for an N processor configuration.

1. INTRODUCTION

Region representation of images play an important role in image processing, VLSI design and computer graphics. Quadtrees have been extensively used as a data structure for their representation[8,9]. Invariably such an application involves processing of hundreds to thousands of scattered/connected regions. Most commonly used algorithms include those for finding the boundary of a region, finding the neighbors of nodes in the tree, and performing operations on these trees.

Parallel computation of geometric properties from the medial axis transform is reported in [12] and parallel processing of linear quadtrees on a mesh-connected computer has been investigated and reported in [3]. Quadtree problems on a two dimensional shuffle exchange network appear in [4,5]. However to the best of our knowledge no parallel boundary following of regions to generate boundary codes has so far been reported in the literature. A scheme for encoding quadtrees in a linear array and algorithms for neighbor finding and boundary following is developed in [1].

In this paper we propose a parallel architecture for following the boundary of regions in an image and provide an adaptation of the boundary following algorithm (BFA) for the proposed parallel architecture. The proposed architecture consists of a fully grown quadtree of processors (PEs) of depth h, where h is less than the depth of the image quadtree. The leaf nodes of the processor tree are also connected as a 2-dimensional mesh. The image quadtree is fully grown upto a depth h. This step is performed in parallel using the PEs of the processor quadtree that are at a depth less than h. After the image quadtree is grown, a subimage can be allotted to every PE of the 2-dimensional mesh. Each processor may get more than one region. Also, some regions may get divided into multiple subregions that get allotted to different PEs. We introduce virtual boundaries whenever the region is divided at the subimage boundaries (alloted to a PE), such that independent subregions can be formed. It is thus possible to follow the boundaries of all regions and subregions allotted to a PE independently. Thus all the PEs perform the boundary following algorithm on the subimages allotted to them in parallel and generate boundary codes.

Having generated the partial boundary codes corresponding to the different subimages we have to combine them to obtain the actual boundary codes of the overall image. Thus an obvious scheme to combine the boundary codes generated by any four PEs which have a common parent PE is to communicate their partial results to their parent PE which suitably combines them. But, such a scheme for combining the partial results is very inefficient. In this paper we present an efficient scheme to combine the partial boundary codes. If an unbounded processor model is assumed the overall parallel algorithm for following boundary of regions stored as an n x n pixel image requires O(logn) time. The sequential time required for boundary following of regions of such an image is O(n). These results are based on the assumption that the average number of regions per unit area is a constant. It may be mentioned that the algorithm is highly scalable in terms of the number of PEs used an can easily be modified for an implementation with fewer PEs.

Consequently a question arises, as to why should one try to develop a sublinear time algorithm for boundary following of regions of an image, when the time required for reading the boundary code itself is linear in n. The answer to the above question lies in the fact that interpreting the boundary code of regions for certain applications does not require reading the entire boundary code. As an example the boundary following of quadtree has been used for certain applications in computer-aided design of VLSI circuits viz. design rule check of VLSI layouts [6,8,9] which use quadtrees as the data structure to store layout descriptions. For such applications interpreting the boundary code requires finding out if the length of any edge in the region boundary is below a threshold value as determined by the design rules. This does not require reading the entire boundary code.

The technical issues of how to partition the image and how to combine partial boundary codes to make the final results in parallel are discussed in section 5. We also carry out an analysis of the parallel BFA and analyse the VLSI layout area complexity of the proposed architecture. For the sake of completeness the scheme adopted for encoding quadtrees in a linear array[7] is reproduced in the following section.

2. ENCODING QUADTREES IN A LINEAR ARRAY[7]

A node in a quadtree is a number called K_value where, K_value=$k_n k_{n-1} \dots k_1$ and $0 <= k_i < 5$. Each digit k_i denotes the path to be taken at level i, (0=terminal 1=NW, 2=NE, 3=SW, 4=SE) to reach the node from root which is assigned a level 0. The width of a K_value is the total number of digits in it. Each element of the linear array storing the quadtree contains 2 bits of information used to represent the colortype of the node as follows.

type colortype=(white,black,gray,not_used);

In the rest of the paper, we adopt the definitions and algorithms given in [2,3]. In addition we define the terms subimage-boundary and corner as shown in fig.2.1. The figure is self explanatory.

3. PARALLEL BOUNDARY FOLLOWING ALGORITHM

The sequential BFA for regions represented in a quadtree appears in [2]. An adaptation of the same algorithm for regions encoded in a linear quadtree can be found in [1]. In this section we present a method to perform the BFA in parallel on the proposed architecture. Consider the image in Fig. 3.1(a). To map this image on to four PEs, connected in a 2-d mesh, we partition the image into four subimages as shown in Fig. 3.1(b). Each subimage is then alloted to individual PEs. If the sequential BFA[2] is mapped as such onto the parallel architecture, then algorithm locates the starting point A (which corresponds to the northernmost black node in the tree) on the boundary of the region of interest in the image. PE(1,1) will traverse the region- boundary from point A until it reaches B. At this stage the PE(1,1) has to pass a message to PE(1,2) to resume the traversal of the region-boundary until point C is encountered. Following this a message is passed to PE(2,2) and the BFA is performed until point D. The above process continues till PE(1,1) receives a message from PE(2,1) and completes following the region-boundary from point E to A. Clearly, such a scheme to follow the boundary of such a region is inherently sequential, and the time required to perform BFA using four PEs as shown above is more than that required by a single PE. The situation further degrades when the boundary of the region crosses the same PE boundary (i.e. the subimage-boundary) an arbitrarily large number of times (Fig. 3.2(a)) or the region spans over many PEs (Fig. 3.2(b)) or if region-boundary to be followed (e.g. the boundary BC) flushes with the subimage-boundary Fig. 3.2(c). Also for regions that are divided into different subregions at the PE boundaries, the BFA will result in synchronization delays. In order to circumvent the above problems, we provide an algorithm for following the boundary of multiple regions in parallel. In the rest of the paper, we adopt the following definitions.

Clipped Region: If an image is partitioned into subimages, (one or more) regions may be divided into subregions at the PE boundaries. We shall refer to such subregions as clipped regions.

Pseudo-boundary: The boundary of a clipped region, running along the subimage-boundary is called pseudo-boundary.

As shown in Fig. 3.1 an actual boundary of the original region may appear as a pseudo boundary of its subregions. The BFA can be performed on all clipped regions alloted to different PEs simultaneously. The boundary code generated for the different subregions is combined to get the overall boundary code for the regions in the original image. Such a scheme, will therefore result in the boundary code being generated for the pseudo boundary also. The boundary codes for the pseudo boundaries are distinguished from the remaining boundary codes and selectively removed while combining the codes of the different subregions.

4. PARALLEL ARCHITECTURE

We store the entire image as a linear quadtree in memory, to be henceforward referred to as **central memory**. This memory is accessible to the different PEs. Individual PEs can address mutually exclusive portions of the memory. In other words different portions of the main memory appear as **local memory** for the different PEs.

The proposed architecture consists of PEs connected as a quadtree, the leaf nodes of which are also connected as a 2-dimensional mesh as shown in Fig. 4.1. Thus in a processor quadtree of depth h, $(4^h-1)/3$ PEs are connected as a quadtree where the PEs at level h-1 (terminal PEs in the processor quadtree) are also connected as a 2-dimensional mesh. Thus there are 4^{h-1} PEs connected as a 2-dimensional mesh of size $2^{h-1} \times 2^{h-1}$. The processors at any level l, $0 \leq l < h-1$, are connected only to PEs at level (l-1) and (l+1) along the quadtree interconnection. These PEs are called **Routers** and have very limited capabilities only to load the quadtree as described later in the procedure load_and_grow_QT.

5. MAPPING THE IMAGE ON TO PROCESSORS

Since the processors are arranged as a quadtree, they can be assigned K_values as described in section 2. Thus $(4^h - 1)/3$ PEs are connected as a quadtree of depth h. Each PE is thus identified by a K_value $K_{h-1}K_{h-2}\ldots K_0$. This K_value is a string of length h. We assume the image size to be m x m pixels where $m = 2^q$ and $0 \leq h < q$. Nodes in the image quadtree are identified by a K_value $K_{q-1}K_{q-2}\ldots K_0$. Further, it is assumed that the image quadtree is virtually grown upto a level (h-1). The nodes of the image quadtree are mapped to the PEs as follows. (i) For all PEs at depth l, $0 < l < h-1$, the node of the image quadtree with K_value $00\ldots0K_{h-1}K_{h-2}\ldots K_0$ is mapped onto the PE with K_value $K_{h-1}K_{h-2}\ldots K_0$. Since, the quadtree is fully grown upto depth h, all nodes bearing K_values $00\ldots0K_{h-1}K_{h-2}\ldots K_0$ exist. Thus the K_value of the node in the image quadtree assigned to any PE is obtained by appending K_value of a PE to a string of (q-h) zeros. (ii) For all PEs at depth h, all the nodes of the image quadtree with K_values denoted by $X_{q-1}X_{q-2}\ldots X_{q-h}K_{h-1}K_{h-2}\ldots K_0$ (where any 'X_i' can assume values 0..4, provided the corresponding nodes exist) are mapped onto the PE with K_value $K_{h-1}K_{h-2}\ldots K_0$. Thus a subtree corresponding to a subimage of size $2^{q-h} \times 2^{q-h}$ pixels is mapped onto every PE at depth h of the processor quadtree.

6. THE ALGORITHM AND ITS COMPLEXITY ANALYSIS

A pseudo Pascal description of the overall algorithm is given below as A6.1.

Algorithm A6.1

```
begin
    load_and_grow_QT;
    for all PEs do in parallel
        begin
            local_BFA;
            Selectively_eliminate_boundary_code;
            Output result in Result_memory;
        end;
end;
```

We assume the image to consist of m x m pixels where $m=2^q$. The processor tree is assumed to be of height h-1 where h=p+1. Thus the PEs at the level (h-1) are connected as a two dimensional mesh. A subimage of size $2^{(q-p)} \times 2^{(q-p)}$ is allotted to each procssor at level (h-1).

6.1 Load_and_Grow_Tree : In this procedure, the image quadtree is virtually developed upto a depth h-1. This is achieved by communicating every Black(White) node assigned to a PE at level l, $0 < l < h-1$, to all its child PEs. Such a step makes available the information of terminal nodes of the quadtree to all PEs at level h-1, without actually growing the tree. This is necessary because the local memory corresponding to the child PEs of all those PEs which have terminals nodes of the image quadtree at level l, $0 < l < h-1$ will contain not used information.

The time required at each level is O(1). Since there are (p+1) such levels, the overall time required for the procedure is O(p).

6.2 Local_BFA : This algorithm performs the BFA on all the regions and clipped regions enclosed by the subimage. The subimage is first enveloped along all its boundaries by white nodes [2. The procedure local BFA is same as the sequential BFA [2]. The boundary code generated during the traversal of the regions are stored in a list. Each boundary_node in the list has a pointer to the next boundary_node. Thus whenever the region boundary crosses PE boundaries, the continuity of the overall boundary code across PEs can be achieved by maintaining pointers appropriately across PE boundaries. Such a pointer in any PE contains the memory address of a location of some other PE.

Since the procedure local_BFA involves boundary following of different regions of the subimage, the time required by any PE is $O(\Sigma p_i)$ where Σp_i is the sum of the perimeters of the regions and the subregions present within the subimage allotted to every PE. If we assume that the number of regions per unit area is constant then $O(\Sigma p_i)$ can also be written as $O(2^{q-p})$.

6.3 Identifying Pseudo Boundary Code : We maintain lists which contain information about points, where Region Boundaries-Intersect Subimage-Boundaries (called RBISB_List). Thus with every boundary d (d can be N, E, S or W) of a subimage we associate the RBISB_List(d).

Every boundary of the subimage has a starting point (called Begin_point) and an ending point (called end_point), corresponding to a clockwise traversal of the subimage-boundary. With every entry in the RBISB_List(d), we also enter the distance of the corresponding intersection point from the starting point of subimage-boundary d. To make an entry in the RBISB_List corresponding to any subimage-boundary d, we have to identify if the region-boundary intersects that subimage-boundary, and if so, then we have to identify if the region-boundary joins the subimage-boundary or forks from it (fig.6.1). This information is stored in the booleans JOIN and FORK respectively. Thus for the entry corresponding to the point B (C) in the RBISB_List(E), the boolean JOIN is TRUE (FALSE) and FORK is FALSE (TRUE). When a region-boundary encounters a corner we make entries in the RBISB_Lists of the subimage-boundaries corresponding to the directions previous_d and opposite(current_d). For example in fig.6.1 the boundary of region R1 meets the NE corner of the subimage at the point B of the region-boundary. At this stage the previous_d becomes E (corresponding to the segment AB) and the current_d becomes S (corresponding to the segment BC). Since the region-boundary AB joins the East subimage-boundary, an entry is made in the RBISB_List(Boundary_of(E)) which has the boolean JOIN set TRUE. Also the region-boundary BC forks from the Boundary_of(N) where N is opposite_dir(S). Thus an entry is also made in RBISB_List(Boundary_of(N)) which has boolean FORK set FALSE.

6.4 Processing RBISB_Lists : Every boundary of a subimage is processed by two PEs so as to perform the following tasks :

1) to find, which part of the pseudo-boundary code lying along the subimage-boundary corresponds to actual boundary code and which part of it has been introduced due to clipping of the regions and does not correspond to the actual boundary code.

2) To classify the points present in the RBISB_Lists as

a) those that enter the PE boundaries, and store the information regarding that point in Enter_PE list of that subimage-boundary b) those which leave the PE boundaries and store the corresponding information in Exit_PE list of that subimage-boundary.

c) those points at which the region-boundary neither enters the PE boundary nor exits it. These points are not stored anywhere. The above tasks are performed by procedure SEPB_Code.

To perform the above tasks we need a list which contains all the intersection points of region-boundary with subimage-boundary for every subimage-boundaries. Such a list is obtained by merging the two RBISB_Lists associated with a particular subimage-boundary. For this, the two PEs which share that boundary of the subimage exchange their RBISB_Lists (associated with that subimage-boundary) and merge them (independently) to obtain a combined intersection list (CI_List) in both the PEs. These lists are processed by the two PEs independently so as to form their respective Enter_PE and Exit_PE lists.

Further we have to process all possible node pairs of the combined intersection list corresponding to pairs of consecutive (or coincident) intersection points present on the subimage boundary. This can best be performed by sorting the CI_List in the increasing order of the distances of the intersection points from the begin_point (associated with the boundary) in both the PEs.

6.5 Forming Enter_PE and Exit_PE Lists : The node pairs from the CI_List (T1,T2) are taken one by one and processed as follows. Let the PE under consideration refer to itself as this_PE and the neighbouring PE as other_PE. Hence every pair of the (T1,T2) is characterised by the following features :

1) Whether T1 belongs to this_PE or other_PE,
2) Whether T2 belongs to this_PE or other_PE,
3) Whether T1 joins or forks at the intersection point.
4) Whether T2 joins or forks at the intersection point.

For any particular value of the above features we can uniquely determine if an actual boundary exists between nodes T1 and T2, and whether the point corresponding to T1 (T2) is an Exit_PE or Enter_PE point. Since each of the features can be characterised by boolean values, there exist in all 16 combinations of the four features, some of which cannot be realised for real images. If D1 and D2 denote the distance of T1 and T2 from the begin_point of boundary E of this PE respectively. Since the intersection points of the CI_list are sorted, either $D2 > D1$ or $D2 = D1$. Thus 16 combinations exist for the case $D2 > D1$ and 16 for the case $D2 = D1$.Actions for individual cases are enumerated in algorithm interpret_region_boundary_intersection in the Appendix and illustrated in fig.6.3.

Together with the processing of the points of the CI_List associated with a boundary, the corresponding Exit_PE and Enter_PE lists are formed. Every node of the Exit_PE or Enter_PE list also contains a Boundary_pointer. For every entry in the Exit-pe list of this PE, there exists a corresponding entry in the Enter_PE list of the other PE. Hence a PE communicates with the neighbouring PEs along its different boundaries, receives the corresponding Enter_PE lists from the respective neighbouring PE and establishes all such pointers for boundary nodes across PE boundaries.

Corner_Dummy Nodes : There may be ambiguity at the corners of the subimage since it is possible to find the continuation of the region boundary in two PEs which also contain that corner. Such ambiguities are resolved using certain dummy nodes at the corners as described below. If an Exit_PE point exists at subimage boundary corner, the corresponding Enter_PE point may appear (Fig.6.1) in Enter_PE lists of two other PEs that also share this corner. In such cases we therefore do not make entries in the Enter_PE and Exit_PE lists. Instead we maintain four dummy nodes at the corners of a subimage. In a clockwise traversal of the subimage boundary, we encounter four corner dummy nodes. Correspondingly for every dummy node we can attribute a boundary that arrives at that corner and another boundary that leaves that corner. If a boundary that arrives at a corner ends in an actual boundary, then the last boundary node of the region-boundary is made to point at the corresponding corner dummy node. On the other hand, if the subimage-boundary that leaves a corner begins with an actual boundary from that corner, then the corresponding corner dummy node is made to point at the first boundary node of the region-boundary. Fig.6.1 illustrates how the corner dummy nodes help in resolving the ambiguities at the corners by establishing proper continuity of boundary code across PEs.

If r represents the number of region boundary crossings across any boundary, the time required for the the sorting step if we use comparision based sorting becomes O(rlogr). It can be seen that a hash based sorting scheme can easily be used to sort the r elements in O(r) time. Since the number of region bundary crossings at any boundary can be no more than 2^{q-p} the time required for the overall procedure is O(rlogr)=O(2^{q-p}(q-p)) for comparision based sorting and O(r)=O(2^{q-p}) for hash based sorting.

The time required for outputting results is proportional to the number of elements in the Exit_pe or the Enter_pe lists which is equal to O(2^{q-p})

The steps 2 to 4 are performed in all the PEs at the level h-1 of the PE tree in parallel. Hence the overall time complexity of the parallel BFA is O(2^{q-p}+p). For values of q=p+logp+O(1), the overall time required for the parallel BFA is O(p). It can also be seen that the time required for the parallel BFA other than the time required for step load_and_grow_QT becomes equal to O(1) for values of q=p+O(1).

An efficient scheme to implement the proposed architecture on VLSI is given below. The area of the layout is analysed in the following manner. Assuming that the quadtree can be stored using N memory locations, each of constant number of bits, such that N=O(2^{2q}) for an image of size 2^q x 2^q . The number of PEs at the level p of the PE tree is M= 2^px2^p. If we assign 2^{q-p}x2^{q-p} pixels per PE, we require O($2^{2(q-p)}$) memory per PE. For the value of q=p+logp+O(1), we have N/log^{2}N PEs each having O(log^{2}N) memory. If we assume the PE area to be dominated by that of the memory the area per PE becomes O(log^{2}N). Since the PEs at the level p of the PE quadtree are connected as a 2-D mesh (Fig. 4.1), overall layout area is = Number of PEs x area per PE = O(N/log^{2}N)xO(log^{2}N) = O(N).

The layout of a quadtree of N routers arranged in a fashion similar to an H-tree requires O(N) area. The total number of routers in our architecture is 4^p/3, which is O(N/log^{2}N). Thus as shown in Fig. 7.1, the layout of the architecture requires O(M) area for M PEs and 4*M/3 routers. It can also be seen that to perform BFA on an image of sqrt(N)xsqrt(N) pixels the architecture requires O(N) area assuming constant area per memory cell.

Acknowledgements : The authors thank K.Sriram and Y.V.S.Prabhakar Rao for their help in preparing the manuscript.

CONCLUSIONS

In this paper we proposed a parallel architecture to follow the boundary of regions represented as a linear quadtree in parallel. Through a complexity analysis of the algorithm, we establish that the overall complexity of the parallel boundary following algorithm is O(log n) for an image of size n x n as compared to O(n) time required on a uniprocessor. This analysis assumes the number of regions per unit area to be constant.

REFERENCES

[1]. Bhaskar, S K, Rosenfeld, A and Wu, A Y "Parallel Processing of Regions Represented by Linear Quadtrees", Computer Vision, Graphics, and Image Processing, 42, 1980,pp. 371-380.

[2]. Dyer, R et al. "Region Representation: Boundary Codes from Quadtrees", Communications of the ACM, Vol. 23, 1980, pp. 171-179.

[3]. Hung, Y and Rosenfeld, A "Parallel processing of linear quadtrees on a mesh-connected computer", CS-TR-1817, University of Maryland, March 1987.

[4]. Mei, G G and Liu, W "Parallel Processing for Quadtree Problems", Proceedings of the International Conference on Parallel Processing, 1986, pp. 895-902.

[5]. Mei, G G and Liu, W "Quadtree Problems on a Two Dimensional Shuffle Exchange Network", Proceedings, Coomputer Vision Pattern Recognition, 1986, pp.140-147.

[6]. Nandy, S K and Deshmukh, S M "Linear Time Geometrical Design Rule Checker based on Quadtree representation of VLSI Mask Layouts", Technical Report, Center for CAD, I.I.Sc, April 1987.

[7]. Nandy, S K, Moona, R and Rajagopalan, S "Linear Quadtree Algorithms on the Hypercube", Proceedings of the 17th annual International conference on Parallel Processing, August 1988 (to appear).

[8]. Nandy, S K and Patnaik, L M "Linear Time Geometrical Design Rule Checker based on Quadtree representation of VLSI Mask Layouts", Computer-Aided Design, Vol. 18, No. 7, September 1986, pp. 380-388.

[9]. Nandy, S K and Ramakrishnan, I V "Dual Quadtree representation for VLSI Designs", Proceedings of the 23rd Design Automation Conference, July 1986, pp. 663-666.

[10].Samet, H "Neighbor Finding Techniques for Images Represented by Quadtrees", Computer Graphics and Image processing, 18, 1882, pp. 37-57.

[11].Samet, H "The quadtree and Related Hierarchical Data Structure", ACM Computing Surveys, 16, 1984, pp. 187-260.

[12].Wu, A Y, Bhaskar, S K and Rosenfeld "Parallel Computation of Geometric Properties from the Medial Axis Transform", Computer Vision, Graphics, and Image Processing, 41, 1988, pp. 323-332.

APPENDIX

The procedure Interpret_region_boundary_intersection(T1,T2) is described with regard to Fig. 6.2(i)-6.2(xvi) wherein the subimage boundary between T1 and T2 is interpreted. T1 and T2 are two consecutive nodes from templist and are the region-boundary intersection points during the clockwise traversal of the boundary of the subimage. Let d1 and d2 denote distance(T1,origin) and distance(T2,origin) such that d1 < d2. We merge the two lists using the distance(T, begin_point) as the first key and the boolean not(This_PE) as the second key assuming FALSE < TRUE. The interpretation corresponding to pair (T1, T2) is unique for any particular boolean assignment for the following predicates associated with T1 and T2, viz.

a). P1 := (T1.Pe_id = this_PE)
b). P2 := (T1.JOIN = TRUE)
c). P3 := (T2.Pe_id = this_PE)
d). P4 := (T2.JOIN = TRUE)

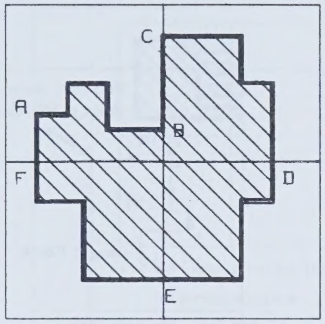
Actions for the individual cases corresponding to the boolean assignments of P1, P2, P3 and P4 are given below. The actions are based on the assumption that, for actual boundaries that lie on subimage boundaries, the corresponding boundary code appears in the processor which contains the region corresponding to that boundary. Based on the boolean values of the four predicates, an intersection point may get classified as an Enter_PE point or an exit_PE point or as none of the two categories.

```
Procedure
Interpret_region_boundary_intersection(T1,T2);
Begin {no op : no operation}
 Case (P1,P2,P3,P4) of
   (F,F,F,F) : if d1 < d2 then Error condition {(i)a }
                    else Error condition;{ CASE (i)b }
   (F,F,F,T) : if d1 < d2 then  /* no op */  {(ii)a}
                    else  Error condition; {(ii)b}
   (F,F,T,F) : if d1 < d2 then
                       Begin
                          Exit_pe_point(T1);
                          Enter_pe_point(T2) {(iii)a }
                       end
                    else ; /* no operation */ {(iii)b}
   (F,F,T,T) : if d1 < d2 then  Exit_pe_point(T2){(iv)a}
                    else  Exit_pe_point(T_new);  {(iv)b}
   (F,T,F,F) : if d1 < d2 then  /* no op */ {(v)a}
                    else Error condition; {(v)b }
   (F,T,F,T) : if d1 < d2 then  Error condition {(vi)a }
                    else Error condition; {(vi)b}
   (F,T,T,F) : if d1 < d2 then Enter_pe_point(T1) {(vii)a}
                    else Enter_pe_point(T_new); { (vii)b}
   (F,T,T,T) : if d1 < d2 then  /* no op */ {(viii)a}
                    else ; /* no op */ {(vii)b}
   (T,F,F,F) : if d1 < d2 then  /* no op */ { (ix)a}
                    else ; /* no op */ { (ix)b}
   (T,F,F,T) : if d1 < d2 then Enter_pe_point(T1) {(x)a}
                    else Enter_pe_point(T_new); { (x)b}
   (T,F,T,F) : if d1 < d2 then  Error condition { (xi)a}
                    else  Error condition; { (xi)b}
   (T,F,T,T) : if d1 < d2 then  /* no op*/ { (xii)a}
                    else Error condition; { (xii)b}
   (T,T,F,F) : if d1 < d2 then Exit_pe_point(T2) {(xiii)a}
                    else  Exit_pe_point(T_new); {(xiii)b}
   (T,T,F,T) : if d1 < d2 then
                       Begin
                          Exit_pe_point(T1);
                          Enter_pe_point(T2) {(xiv)a}
                       End
                    else; /* no operation */ { (xiv)b}
   (T,T,T,F) : if d1 < d2 then /* no op */ {(xv)a}
                    else Error condition; {(xv)b}
   (T,T,T,T) : if d1 < d2 then Error condition {(xvi)a}
                    else Error condition; { (xvi)b}
 End
End;
```

TABLE I

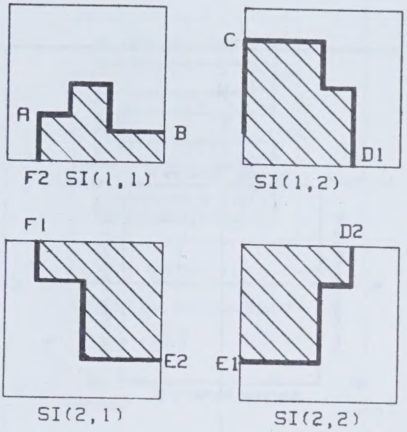
(image size : n x n pixels and processor mesh : m x m at level h-1)

ITEM	COMPLEXITY wrt. m as a parameter	COMPLEXITY for m=n/(k*logn) k ≥1	COMPLEXITY for m=n/k k≥1
1. Perimeter of subimage per processor.	O(n/m)	O(logn)	O(1)
2. Time with normal sort for:			
a. Preprocessing step, (Virtually_grow_QT)	O(logm)	O(logn)	O(logn)
b. BFA (finding boundary codes)	O(n/m*log(n/m))	O(lognloglogn)	O(1)
3. Time with hashing based sort for:			
a. Preprocessing step, (Virtually_grow_QT)	O(logm)	O(logn)	O(logn)
b. BFA (finding boundary codes)	O(n/m)	O(logn)	O(1)
4. Router characteristics			
a. No. of routers	$4m^2/3$	$O(n^2/\log n)$	$O(n^2)$
b. No. of links per router	5 links per processor (4 links to child processors and 1 to parent)		
c. Memory per router	O(1)	O(1)	O(1)
5. Processor characteristics			
a. No. of processors	m^2	$O(n^2/\log^2 n)$	$O(n^2)$
b. No. of links per processor	5 links per processor (4 links to neighbors and one to parent)		
c. Memory per processor	$O(n^2/m^2)$	$O(\log^2 n)$	O(1)



A region (A,B,C,D,E,F) partitioned into four subimages

(a)


A
B
F2
SI(1,1)
C
D1
SI(1,2)
F1
D2
E2
E1
SI(2,1)
SI(2,2)

Subimage SI(i,j) allotted to PE(i,j)

(b)

Fig. 3.1

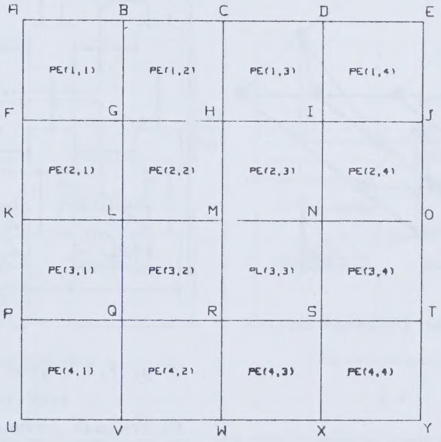

A
B
C
D
E
PE(1,1)
PE(1,2)
PE(1,3)
PE(1,4)
F
G
H
I
J
PE(2,1)
PE(2,2)
PE(2,3)
PE(2,4)
K
L
M
N
O
PE(3,1)
PE(3,2)
PL(3,3)
PE(3,4)
P
Q
R
S
T
PE(4,1)
PE(4,2)
PE(4,3)
PE(4,4)
U
V
W
X
Y

fig. 6.2 Mapping of the Subimages on to 2-D Array of Processors

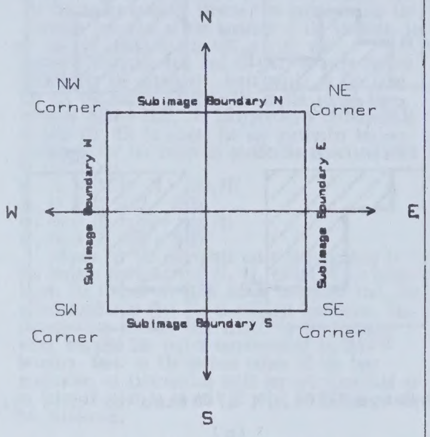

N
NW Corner
Subimage Boundary N
NE Corner
Subimage Boundary W
Subimage Boundary E
W
E
SW Corner
Subimage Boundary S
SE Corner
S

Fig. 2.1 Subimage Boundaries Corresponding to the Four Directions

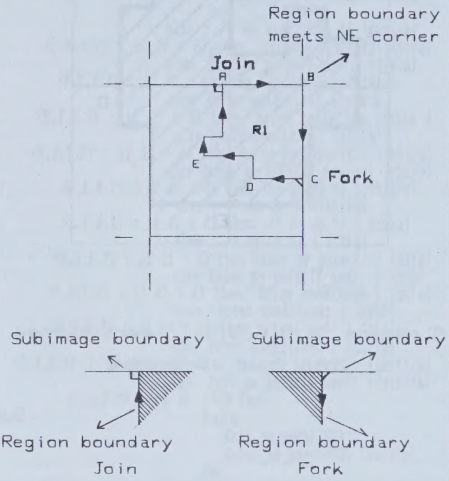

Region boundary meets NE corner
Join
A
B
R1
E
C
Fork
D
Subimage boundary
Region boundary
Join
Subimage boundary
Region boundary
Fork

Fig. 6.1 A region illustrates join and fork

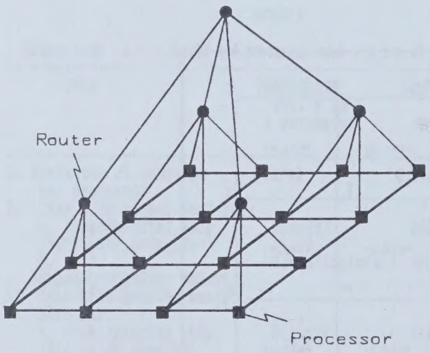

Router
Processor

Fig. 4.1 Processor Interconnection

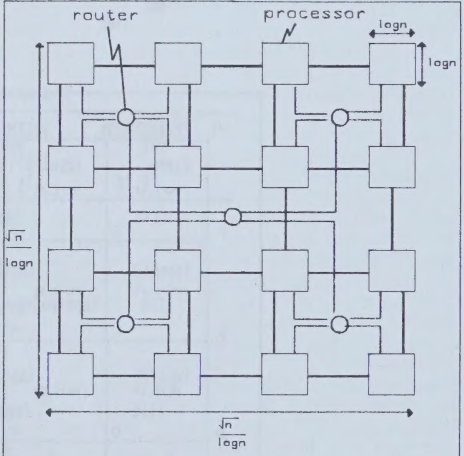
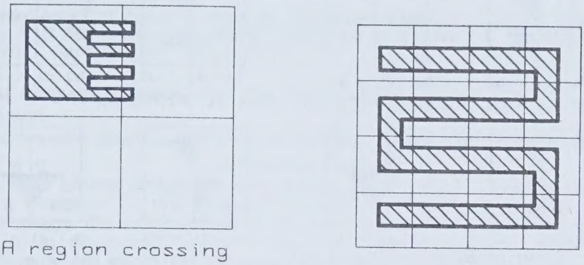

router
processor
logn
logn
$\frac{\sqrt{n}}{\log n}$
$\frac{\sqrt{n}}{\log n}$

Fig. 7.1 Layout of the proposed architecture to process images of n x n pixels



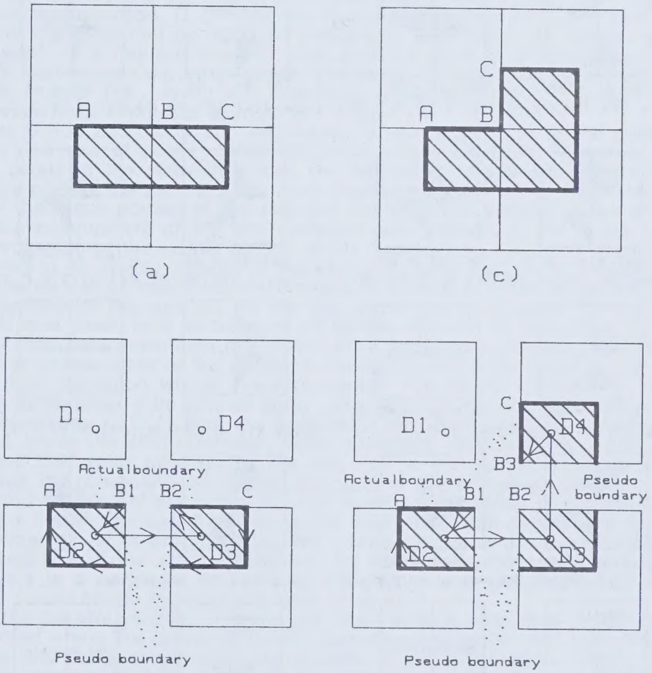
A region crossing a processor boundary several times

(a)

A region extending over several processors

(b)

Fig. 3.2


A
B
C
C
A
B
D1
D4
Actualboundary
A
B1
B2
C
D2
D3
Pseudo boundary
D1
C
D4
B3
Actualboundary
A
B1
B2
Pseudo boundary
D2
D3
Pseudo boundary

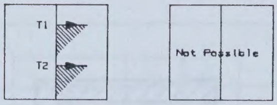
(a)

(c)

(b)

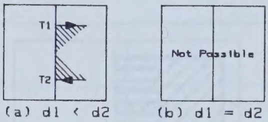
(d)

Fig 6.1 Use of boundary nodes to resolve ambiguities at corners for continuation of boundary code


T1
T2
Not Possible

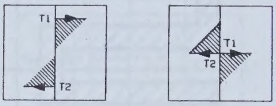
(a) d1 < d2 (b) d1 = d2

Case I (P1,P2,P3,P4) = (F,F,F,F)


T1
T2
Not Possible

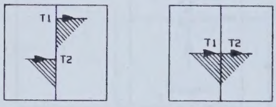
(a) d1 < d2 (b) d1 = d2

Case II (P1,P2,P3,P4) = (F,F,F,T)



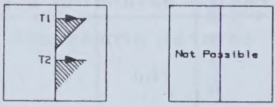
(a) d1 < d2 (b) d1 = d2

Case III (P1,P2,P3,P4) = (F,F,T,F)



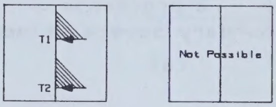
(a) d1 < d2 (b) d1 = d2

Case IV (P1,P2,P3,P4) = (F,F,T,T)


T1
T2
Not Possible

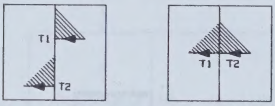
(a) d1 < d2 (b) d1 = d2

Case V (P1,P2,P3,P4) = (F,T,F,F)


T1
T2
Not Possible

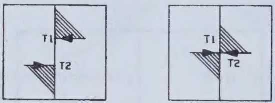
(a) d1 < d2 (b) d1 = d2

Case VI (P1,P2,P3,P4) = (F,T,F,T)



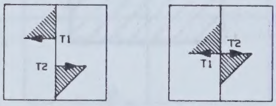
(a) d1 < d2 (b) d1 = d2

Case VII (P1,P2,P3,P4) = (F,T,T,F)



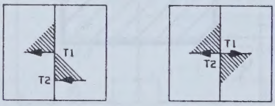
(a) d1 < d2 (b) d1 = d2

Case VIII (P1,P2,P3,P4) = (F,T,T,T)



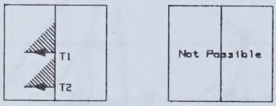
(a) d1 < d2 (b) d1 = d2

Case IX (P1,P2,P3,P4) = (T,F,F,F)



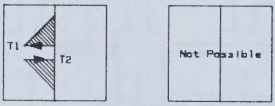
(a) d1 < d2 (b) d1 = d2

Case X (P1,P2,P3,P4) = (T,F,F,T)


T1
T2
Not Possible

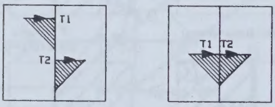
(a) d1 < d2 (b) d1 = d2

Case XI (P1,P2,P3,P4) = (T,F,T,F)


T1
T2
Not Possible

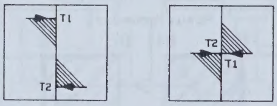
(a) d1 < d2 (b) d1 = d2

Case XII (P1,P2,P3,P4) = (T,F,T,T)



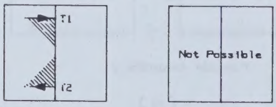
(a) d1 < d2 (b) d1 = d2

Case XIII (P1,P2,P3,P4) = (T,T,F,F)



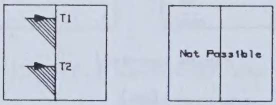
(a) d1 < d2 (b) d1 = d2

Case XIV (P1,P2,P3,P4) = (T,T,F,T)


T1
T2
Not Possible

(a) d1 < d2 (b) d1 = d2

Case XV (P1,P2,P3,P4) = (T,T,T,F)


T1
T2
Not Possible

(a) d1 < d2 (b) d1 = d2

Case XVI (P1,P2,P3,P4) = (T,T,T,T)

Fig. 6.3 Interpretation of Regions for Intersection Pair (T1, T2)

OPTIMAL VISIBILITY ALGORITHMS FOR BINARY IMAGES ON THE HYPERCUBE

FRANK DEHNE
Center for Parallel and Distributed Computing, School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. Research partially supported by the Natural Sciences and Engineering Research Council of Canada under Grant A9173.
QUOC T. PHAM
Dept. 7H65, Bell-Northern Research, P.O. Box 3511, Ottawa, Canada K1Y 4H7.
IVAN STOJMENOVIC
Department of Computer Science, University of Ottawa, Ottawa, Canada K1N 9B4.

1. INTRODUCTION

Consider a n x n binary image π with pixels $\pi(i)$, $1 \le i \le n^2$, numbered in row-major ordering. Given a direction D, the *parallel visibility* problem consists of determining for each pixel of the image the portion that is visible (i.e., not obstructed by any other black pixel of the image) in direction D from infinity. A related problem, referred to as *point visibility*, is to compute for each pixel the portion that is visible from a given point p.

In this paper, we improve the results in [2] and derive 0(log n) time SIMD algorithms for these two problems on the hypercube, where processor P(i) is assigned to pixel $\pi(i)$, $1 \le i \le n^2$. Since the worst case communication distance of two processors in a n^2-processor hypercube is 2 log n, it follows that both algorithms are asymptotically optimal.

2. PARALLEL VISIBILITY

The basic geometric idea for the algorithm for parallel visibility is to divide the image π into strips parallel to direction D, where each strip is the portion of light going through the top edge of a pixel in the top row of the image (as if it were unobstructed by any black pixel). We shall assume without loss of generality that the angle β between the North-South direction and the direction D (in counter-clockwise direction) is between 0 and 45^o; otherwise, the algorithm can be obtained similarly by symmetry. Let w_0 and $w = \cos(\beta) w_0$ denote the width of a (square shaped) pixel and the width of a strip, respectively. Since $0 \le \beta \le 45^o$, $(\sqrt{2}/2) w_0 \le w \le w_0$; this yields **Property 1** ([1]): *No pixel $\pi(i)$ is properly contained in a strip (i.e., every $\pi(i)$ intersects either the left or the right border of its strip) and every pixel intersects at most two strips.*

We further divide each strip into *segments,* where a segment is the portion of a strip contained in one row of pixels. Since $0 \le \beta \le 45^o$, each segment intersects at most two neighboring pixels in the respective row; the leftmost of these will be referred to as the *representative pixel* of the segment. For each segment, we define the *black interval* to be the projection of the black portion of the segment on the cross-section of the strip; the *white interval* is the complement of the black interval (with respect to the cross-section of the strip). The projection of the visible portion of the segment on the cross-section of the strip will be referred to as *visible interval* of the segment. Note that, from Property 1 it follows that each white as well as visible interval consists of at most one connected component.

In the remainder of this section, we will first show how to compute for each segment its white interval and, then, how to compute all visible intervals in time O(log n). We employ two O(log n) time data movement operations on a hypercube defined in [3]: *distribute* and *concentrate.* A special case of the distribute operation, which can be executed in the same time, is the *shift* operation where every processor P(i) sends a record to the processor P(i+k) in the same row. With this, all white intervals can be computed as follows: every processor determines in O(1) time the segment it represents and the corresponding strip number; then, using the shift operation, every processor examines its local neighborhood and computes the white interval for the segment it represents in O(log n) time. The geometric idea for computing all visible intervals is the following: Consider, within each strip, the sorted ordering of the segments with respect to direction D such that the topmost segment is the first in this ordering. The visible interval of each segment is the intersection of the white intervals of its predecessors (the visible interval of the topmost segment is the entire cross-section of the strip). Therefore, for each strip, the problem of computing all visible intervals is a particular instance of the partial sum (or parallel prefix) problem which can be solved on the hypercube in time O(log n).

To solve the parallel visibility problem, for each strip in parallel a partial sum problem has to be solved where the operands for the partial sum operation are the white intervals of the segments and the associative binary operator is set intersection. However, the partial sum algorithm assumes that the operands for each partial sum problem are stored in exactly one sub-hypercube which, in general, is not the case for the white intervals of a strip. We observe that the processors which store a row or column of pixels form a sub-hypercube. Therefore, our strategy is to move (using the shift operation) each of the strips into a column-subcube of the hypercube so that the partial sums, for all strips, can be computed independently in O(log n) time. Finally the obtained visibility information is returned to the original segment locations and propagated to the neighboring pixel in the segment.

Theorem 1: *The parallel visibility problem for a digitized image of size nxn can be solved on a d-dimensional hypercube, $2^d = n \times n$, in time O(d)=O(log n).*

3. POINT VISIBILITY

In order to determine the visibility from a point p, we will assume without loss of generality that the point p is located at the upper left corner of the image; otherwise, the image can be split by the horizontal and the vertical lines through p into (at most) four quadrants and the problem can be solved for each quadrant separately. In the remainder of this section we will show how to compute for all pixels in the area below the 45^o ray emanating from p (again, all angles are defined with respect to the north-south axis and in counter-clockwise direction) the portion that is visible from p. For all pixels above the ray, the visibility problem can be solved in a second analogous step. Consider the 22.5^o ray emanating from p. It splits the image (below the 45^o ray) into two strips whose widths (i.e., the horizontal distance between left and right border) increase with the distance from p and will, eventually, exceed width w_0 which ensured that no pixel is properly contained in a strip (cf. Property 1). At the level where the width of the rightmost strip reaches w_0, each strip is split again (by rays emanating from p) into two strips such that the angles between the borders of the four strips are equal. When the width of the rightmost of these four strips reaches w_0, they are bisected again. This process is repeated until the entire image below the 45^o ray is covered. We define a *sector* to be the section of a strip between two consecutive splittings. For every horizontal cut through the image, the width w_l of the leftmost sector intersected by the cut has the property $w_r > w_l = (1+\tan\alpha/2)w_r \geq (1/2)w_r$, where w_r is the width of the rightmost sector (intersected by the cut) and α the angle between the left and right border of the sector ($0^o < \alpha \leq 22.5^o$). Since $(1/2)w_0 \leq w_r \leq w_0$, and the width of all other sectors on the cut is between w_l and w_r, we obtain for the horizontal width w of any sector (along any horizontal cut) $(1/4)w_0 \leq w \leq w_0$. This yields

Property 2: *No pixel is properly contained in a sector (i.e., every pixel intersects either the left or right border of its sector) and every pixel intersects at most four sectors.*

Furthermore, we divide each sector into *segments*. A segment of a sector S is the portion of S contained in one row of pixels. For each segment, the *representative pixel* and the *white interval* are defined analogously to the parallel visibility problem. The sectors, together with the relation "<" defined by "$S_1 < S_2$ *if and only if the top horizontal border of* S_2 *is contained in the bottom horizontal border of* S_1", form a binary tree which we will refer to as *sector tree*. The point visibility problem can be solved by executing for every path from the root to a leaf of the sector tree, for the white intervals of the segments of these sectors, the partial sum operation. Compared to Section 2, the problem arising here is not only that segments involved in a partial sum operation are (in general) not stored in a sub-hypercube but also that segmfor the solution of this problem is to copy, for every path from the root to a leaf of the sector tree, the segments of these sectors into a column sub-hypercube. In order to do this efficiently, we utilize the *generalize* operation of [3]. The point visibility problem can now be solved as follows (To simplify exposition, we shall assume that every processor representing a pixel simulates four 'virtual' processors, one for each of the at most four segments intersected by the pixel.):

(1) Every processor P(i) determines in constant time which segment, *Seg(i)*, in which sector, *Sect(i)*, it represents (if any). (2) Since the sector tree is a complete binary tree, every P(i) can also determine in constant time the column number, *c(i)*, of the representative pixel of the bottommost segment in the rightmost leaf-sector of the sub sector tree rooted at Sect(i). If Sect(i) is a leaf-sector then c(i) is the column number of the representative pixel of the bottommost segment of Sect(i). (3) Every P(i) determines in time O(log n) the white interval, *w(i)*, of Seg(i); see Section 2. (4) A generalize operation is performed where, for every P(i) representing a segment Seg(i), the record to be sent is w(i) and the destination address is the row major index, i^*, of the pixel with the same row number as $\pi(i)$ and column number c(i). (It is easy to see that the requirement $i<j \Rightarrow dest(i)<dest(j)$ for generalize operations holds; see [3].) (5) After Step 4, for each path from the root to a leaf of the sector tree, the white intervalls of the segments of all sectors on the path are stored in a column subcube in sorted order. Therefore, for all paths in parallel, the partial sum operation with respect to their white intervals can be computed in time O(log n). (6) The visible intervals obtained in Step 5 are returned from each $P(i^*)$ to P(i), the processor storing the representative pixel of the segment. (This can be implemented with time complexity O(log n) by using the concentrate operation.) Note that, in Step 5, for all copies of the white interval of a segment the result of the partial sum operation is the same. Finally (by using the shift operation), for each segment the visible interval is sent in time O(log n) from the representative pixel to the other pixel in its segment (if exists).

Theorem 2: *The point visibility problem for a digitized image of size n x n can be solved on a d-dimensional hypercube,* $2^d = n \times n$, *in time O(d)=O(logn).*

REFERENCES

[1] F.Dehne, A. Hassenclover, J.-R. Sack, N. Santoro, Parallel Visibility on a Mesh-Connected Computer, in: Proc. Int. Conference on Parallel Processing and Applications, L'Aquila, 1987, pp. 173-180.

[2] F.Dehne, Q.T.Pham, Visibility Algorithms for Binary Images on the Hypercube and the Perfect-Shuffle Computer, in: Proc. of the International Federation for Information Processing WG 10.3 Working Conference on Parallel Processing, Pisa, 1988.

[3] D. Nassimi, S. Sahni, Data Broadcasting in SIMD Computers, *IEEE Transactions on Computers*, Vol. C-30, No. 2, 1981, pp.101-106.

A VLSI PARALLEL PROCESSOR FOR BICUBIC SURFACE PATCH IMAGE GENERATION

PHILIP C. CHAO and MING-YANG CHERN
AT&T Bell Laboratories, Naperville, IL 60566

ABSTRACT- This paper presents a parallel processor for generating ray tracing images of bicubic surface in Bezier form nby using the subdivision algorithm. It uses a set of tree transverse operators to find the "nearest" intersection between a surface patch and a ray by traversing the entire subdivision tree in preorder. This scheme retains the advantage of subdivision and substantially reduces the circuit area by eliminating the need for a stack which is used to store control points of intermediate bicubic patches. The time to traverse additional nodes is offset by the removal of the stack memory access time. The performance of the architecture is analyzed and compared with the alternative approaches.

INTRODUCTION - Parametric surfaces are a popular form of surface and object representation. In particular, bicubic patches have gained popularity for computer graphics and solid modeling systems. A surface formed by joining multiple bicubic patches has the advantage that higher order continuities can be maintained across the entire surface. Because of the popularity of the bicubic patch, accurate and rapid displaying of such surfaces has been a goal of many researchers, for example [1] [5]. Ray tracing is a technique for generating images of three dimensional objects with a computer. Programs using ray tracing can simulate the effects of reflection, refraction, and shadows to produce computer images which possess a strikingly high degree of realism [3] [10]. Central to the ray tracing algorithm is the ray surface intersection calculation. For bicubic patch, it is generally done in one of three ways. First, the surface is approximated with a collection of polygons and ray tracing is performed on the resulting collection of planar surfaces. However, to reduce the effects of polygonal silhouette and discontinuity at the patch boundary, large numbers of polygons are usually required for a good polygon approximation of the patch. The second method is to solve the ray surface intersection directly with some numerical methods such as Newton iteration [4] [9]. Some of these numerical methods become complicated near degeneracies. The third method, the subdivision algorithm, recursively subdivides the patch until reaching the pixel size or matching certain terminating conditions [2] [6] [8]. The "nearest" intersection point between a ray and the bounding box of a terminating subpatch is the visible surface point for the pixel.

Because of the regularity of the subdivision method, it is suitable for VLSI implementation [7]. Nevertheless, there are two drawbacks in this method: (1) Each ray bounding-box intersection computation requires six multiplications, (2) The algorithm requires a stack which stores 48 coordinates for each entry. This consumes a considerable amount of silicon area and access time. In Pulleyblank's VLSI architecture for subdividing a Bezier patch [7], a "subdivision chain" is proposed to reduce the depth of the stack by saving the control points at some selected levels in the tree. However, this scheme does not help much, since: (1) it needs to regenerate intermediate patches multiple times because these patches are not stored in the stack, and (2) the scheme still consumes 50% of the circuit area for its stack memory.

CONCEPT AND ARCHITECTURE - This paper presents an architecture to perform subdivision algorithm by using a set of tree transversal operators without stack. Figure 1.a shows the subdivision operation. From this figure, it can be seen that among the three sets of points : P0-P3, L0-L3, and R0-R3, any two sets of points can be generated from the third set. (See Figures 1.b and 1.c.) All multiplications inside the subdivision iteration are eliminated by dividing the coordinates of the control points of the original patch by the corresponding coordinate of the directional vector of the ray before the iteration begin.

A Shift-Add Add-Shift (SAS) is used as an universal building block to perform the subdivision and reverse subdivision operations. (See Figure 2.) Six SASs and two multiplexers form a Curve Subdivision Block (CSB) which performs all the functions described in the Figures 1.a, 1.b and 1.c. (See Figure 3.) The multiplexer block is capable of reversing the order of the control points when enabled. In order to traverse in the surface patch subdivision tree, we define four tree transverse operators. (See Figure 4.) These operators are Down (D), Up (U), Next (N), and Previous (P). To compute these operators, a Patch Subdivision Block (PSB) is constructed by interconnecting eight CSBs as shown in Figure 5. In the PSB, the top four CSBs and the bottom four CSBs operate in two separate modes. Inputs to the PSB are the 16 coordinates of the control points of a Bezier surface patch, and the output of the PSB are the 16 coordinates of the control points of the resulting patch which may be a parent, a child or a sibling of the input patch. The way to control the relationship between PSB's input and output is to program the SAS and the four multiplexes. Figure 6 shows the 13 configurations of PSB for executing the tree transverse operators defined in Figure4. The label "R" indicates that the multiplexer is in enabled state.

The overall processor architecture for the algorithm is shown in Figure 7. It consists of a PSB, a ray-bounding box intersection circuit (BIC), an IP register, and 3 sets of comparators (CMPs). The IP register stores the line parameter of the currently "nearest" intersection point. The purposes of the 3 sets of comparators are (1) to evaluate the lower bound (LB) and the higher bound (HB) of the line parameters of the intersection points, (2) to compare the HB and LB with the content of the IP register to determine whether a subtree should be pruned or not, and (3) to compare the size of the bounding box with the size of the pixel to decide whether the subpatch is small enough to be a leaf node or not. If the subpatch is not a leaf node, further subdivision will be performed. If the subpatch is a leaf node, two things will happen : the IP register will be updated with the intersection point which has the smallest positive line parameter, and the tree transverse operators will be applied again until the root node is reached.

PERFORMANCE ANALYSIS - Let β be the branching factor of the subdivision tree. Four approaches are compared in our analysis. They are the proposed architecture (i.e., approach 1), the standard depth-first search (with a stack) with and without subdivision chain (approach 2 and 4), and Pulleyblank's approach mentioned above [7] (approach 3). Approach 3 generates four subpatches simultaneously and stores all the subpatches that intersect with the ray into the stack. Figures 8 and 9 show the total processing time and the stack memory area needed for each alternative assuming that 200ns is required to process each patch and an additional 200ns is required to access the 48 coordinates of the control points. The result shows that our proposed approach is the fastest among the four approaches. From the previous discussion, the hardware cost required for our approach is also the lowest.

CONCLUSION - Our proposed parallel processor eliminates the multiplication steps originally needed in the subdivision iteration. It does not require the stack; as a result, it saves stack access time and memory area. It is suitable for VLSI implementation.

REFERENCE

[1] P. Bezier, "Mathematical and Practical Possibilities of UNISURF," in R. E. Barnhill and R. F. Riesenfeld (eds.), Computer Aided Geometric Design, Academic, New York, 1974.

[2] E. Catmull, "A Subdivision Algorithm for Computer Display of Curved Surfaces," UTEC-74-133, University of Utah, 1974.

[3] R. L. Cook, T. Porter, L. Carpenter, "Distributed Ray Tracing," Computer Graphics (Proc. SIGGRAPH 84), vol.18 No.3, July 1984. pp. 137-145.

[4] J. I. Joy, M. N. Bhetanabhotla, "Ray Tracing Parametric Surface Patches Utilizing Numerical Techniques and Ray Coherence," SIGGRAPH 1986, Volume 20, Number 4. pp. 279-285.

[5] J. T. Kajiya, "Ray Tracing Parametric Patches," Computer Graphics, Volume 16, Number 3, July 1982. pp. 245-271.

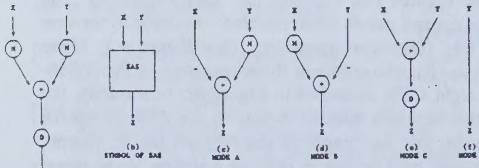
[6] J. M. Lane, R. F. Riesenfeld, "A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces," IEEE Trans. on Pattern Analysis and Machine Intelligence, Jan. 1980, pp.35-46.

[7] R. Pulleyblank, J. Kapenga, "The Feasibility of a VLSI Chip for Ray Tracing Bicubic Patches," IEEE CG&A, March 1987. pp.33-44.

[8] M. A. J. Sweeney, R. H. Bartels, "Ray Tracing Free-Form B-Spline Surfaces," IEEE CG&A, February 1986. pp. 41-49.

[9] D. L. Toth, "On Ray Tracing Parametric Surfaces," SIGGRAPH 1985, Volume 19, Number 3. pp. 171-179.

[10] T. Whitted, "An Improved Illumination Model for Shaded Display," Communications of the ACM, Volume 23, Number 6, June 1980. pp. 343-349.



(a) FUNCTION PERFORMED BY SAS (b) SYMBOL OF SAS (c) MODE A (d) MODE B (e) MODE C (f) MODE X

Figure 2. SHIFT-ADD ADD-SHIFT BLOCK (SAS)

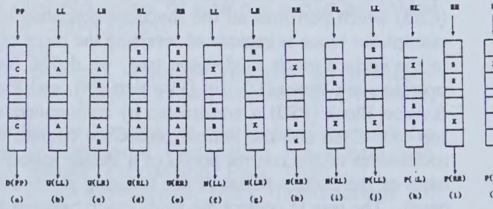


Figure 6. TREE TRANSVERSE OPERATIONS PERFORMED BY PSB

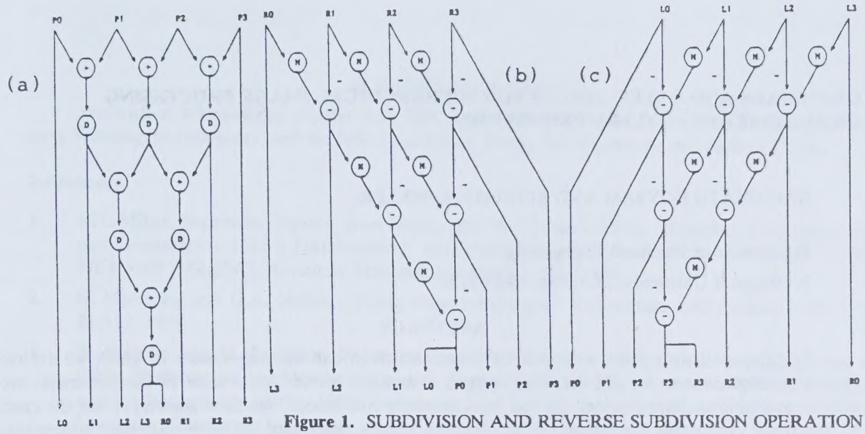
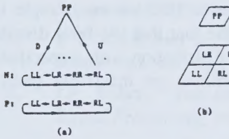

(a)
(b)
(c)
P0
P1
P2
P3
R0
R1
R2
R3
L0
L1
L2
L3
D
N
L0 L1 L2 L3 R0 R1 R2 R3
L3 L2 L1 L0 P0 P1 P2 P3
P0 P1 P2 P3 R3 R2 R1 R0

Figure 1. SUBDIVISION AND REVERSE SUBDIVISION OPERATION


PP
D
U
N:
LL→LR→RR→RL
P:
LL←LR←RR←RL
(a)
PP
LR
RR
LL
RL
(b)

DOWN OPERATOR : D(PP) = LL

UP OPERATOR : U(LL) = U(LR) = U(RR) = U(RL) = PP

NEXT OPERATOR :
N(LL) = LR
N(LR) = RR
N(RR) = RL
N(RL) = LL

PREVIOUS OPERATOR :
P(LL) = RL
P(RL) = RR
P(RR) = LR
P(LR) = LL

Figure 4.
TREE TRANSVERSE OPERATORS

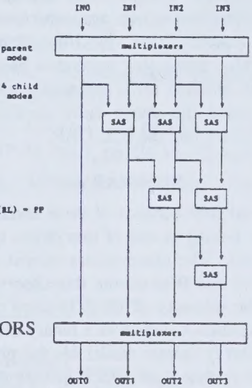

IN0
IN1
IN2
IN3
parent node
4 child nodes
multiplexers
SAS
multiplexers
OUT0
OUT1
OUT2
OUT3

Figure 3. CURVE SUB DIVISION BLOCK (CSB)

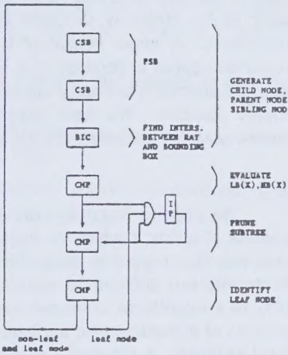

CSB
CSB
BIC
CHP
CHP
CHP
I F
PSB
GENERATE CHILD NODE, PARENT NODE, SIBLING NODE
FIND INTERS. BETWEEN RAY AND BOUNDING BOX
EVALUATE LB(X),RB(X)
PRUNE SUBTREE
IDENTIFY LEAF NODE
non-leaf and leaf node
leaf node

Figure 7. THE PROCESSOR ARCHITECTURE

Figure 5.
PATCH SUBDIVISION BLOCK (PSB)

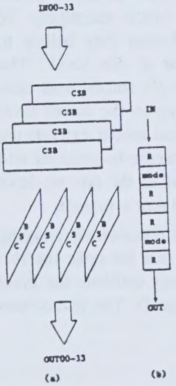

IN00-33
CSB
CSB
CSB
CSB
C S B
IN
R
mode
R
R
mode
R
OUT
OUT00-33
(a)
(b)

Figure 8. TOTAL PROCESSING TIME

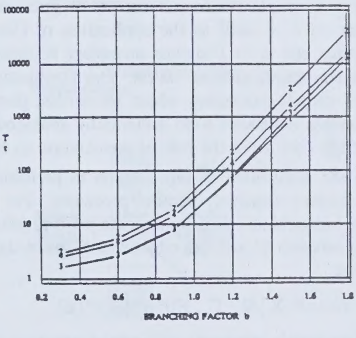
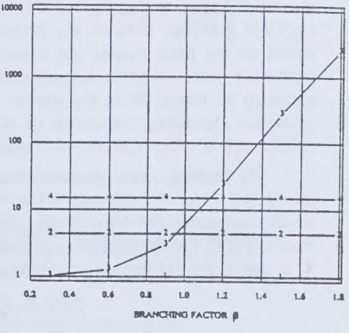


Figure 9. STACK MEMORY AREA



GRAMMARS AND BAYES' PRIORS FOR HIERARCHICAL IMAGE PROCESSING ON MASSIVELY PARALLEL PROCESSORS

BADRINATH ROYSAM AND MICHAEL I. MILLER

Department of Electrical Engineering
Washington University, St. Louis, MO 63130.

ABSTRACT

In this paper, we present a three-level hierarchical Bayesian imaging system in which we exploit constraints applicable to the objects being imaged at multiple levels. While some of the constraints are directly available as Bayes' priors, the rest are essentially rule-based. We have shown [1] that the class of rule-based constraints corresponding to Chomsky regular grammars [2] have equivalent representations in the form of Gibbs' distributions. We use this result to map the grammatical constraints on each layer in the hierarchy to Gibbs distributions which are subsequently used in a Bayesian estimation framework. A major benefit of the representation via Gibbs distributions is that we can sample the constraint-spaces in parallel. Our parallel processing method is based on the fact that the limit distribution of Langevin's stochastic diffusion equation [3] is essentially a Gibbs' distribution with a specifiable energy function. We have implemented the above fully parallel algorithms on massively parallel mesh-connected machines: the NCR GAPP and the ICL DAP.

SUMMARY

The imaging hierarchy considered here consists of three levels. (see Figure 1) (i) The lowest level consists of pixels **X** which we assume belong to one of two object types 0 and 1. The extension to more than two object types is straightforward. The observations consist of realizations from Poisson random fields with two different intensities. We use **D** to denote this observed data. We have used Good's prior [45] as a smoothness constraint on the intensity of these Poisson random fields. (ii) The middle level consists of a characteristic attribute function **C** which is a binary array (Figure 2) that may we denote as pixel attributes. A value of 1 at position ij selects model M_1 for pixel ij while a value of 0 selects M_0. A very simple grammar is applicable at this layer which forbids the value of **C** from changing within an object. This grammatical constraint is disabled when the pixels in question straddle an object boundary. (iii) The third level of the hierarchy consists of edge-sites (figure 3), which essentially lie between pixels and serve to either connect or disconnect pixels depending on whether they belong to the same object or otherwise. We apply a simple run-length constraint grammar at this level. The edge-site grammar controls the grammar on **C** as well as the application of Good's smoothness constraint on the pixel values. An active edge site $e_{ij} = 1$ signals an object boundary to the lower level grammars which switches the applicable constraints at those levels. Our Hierarchical model extends the approach of Baker [6] in the context of speech processing where phonemes, phoneme-boundaries and grammars expressing constraints on phoneme sequences form a hierarchy analogous to the one we have defined above. For example, note that edge sites serve the role of punctuation in Baker's system.

We present results demonstrating the application of this method to perform constrained imaging on the ICL-DAP (Distributed Array Processor) massively parallel processor. For this we pose the following estimation problem. In the above hierarchical image model, we wish to jointly estimate the pixel values **X**, the the characteristic attribute function **C** and the edge sites **E** given data **D**. The parameters **X**, **C** and **E** are chosen so as to maximize:

$$P(\mathbf{X},\mathbf{D},\mathbf{C},\mathbf{E}) = P(\mathbf{X} \mid \mathbf{D}, \mathbf{C}, \mathbf{E}) P(\mathbf{C} \mid \mathbf{E}) P(\mathbf{E})$$

The maximization is performed using Langevin's stochastic differential equation and simulated annealing [7].

ACKNOWLEDGEMENTS

We wish to acknowledge support from NIH grant RR01380 via the Biomedical Computer Laboratory, Washington University, and the NSF Presidential Young Investigator Award ECE-8552518.

References

1. M.I. Miller, Badrinath Roysam, Kurt Smith, and Jan Tyman Udding, "Mapping Rule-Based Regular Grammars to Gibb's Distributions," *AMS-IMS-SIAM Joint Conference on SPATIAL STATISTICS AND IMAGING*, American Mathematical Society, July 1988.
2. N. Chomsky and G.A. Miller, "Finite State Languages," *Information and Control*, vol. 1, pp. 91-112, 1958.
3. B. Roysam and M. I. Miller, "Mapping Deterministic Rules to Stochastic Representations via Gibbs Distributions on Massively Parallel Analog Networks: Application to Global Optimization," *Proceedings of the 1988 Summer School on Connectionist Models*, Carnegie Mellon University, Pittsburgh, June 1988.
4. B. Roysam, J. A. Shrauner, and M. I. Miller, "Bayesian Imaging Using Good's Roughness Measure-Implementation on a Massively Parallel Processor," *Int. Conf. Acoustics, Speech and Sig. proc., IEEE*, New York, March 1988.
5. I. J. Good and R. A. Gaskins, "Nonparametric roughness penalties for probability densities," *Biometrika*, vol. 58,2, pp. 255-277, 1971.
6. J.K. Baker, "The DRAGON System - An Overview," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-23, No.1, pp. 24-29, February, 1975.
7. S. Geman and C.-R. Hwang, "Diffusions for Global Optimization," *SIAM J. Control and Optimization*, 1987.

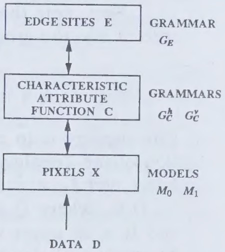


Figure 1: Showing the hierarchical image model.

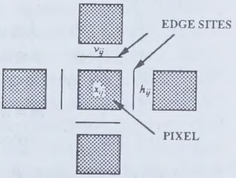


Figure 3: Illustrating the edge-site punctuations

$$C_{ij} = \{ \begin{array}{l} 1 \ \ if \ x_{ij} \in object\ 1 \\ 0 \ \ if \ x_{ij} \in object\ 0 \end{array}$$

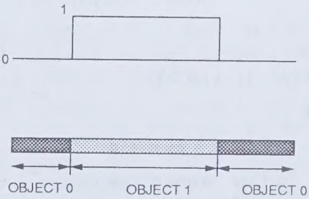


Figure 2: Illustrating the Characteristic Attribute Function

A ROTATION BASED ALGORITHM FOR FAST LEAST SQUARE DECONVOLUTION

Yu Hen Hu and Paul H. Milenkovic***

Department of Electrical and Computer Engineering
University of Wisconsin - Madison
Madison, WI 53706

ABSTRACT

An efficient computation algorithm, based on unitary Jacobi-type rotations, is developed in this paper for fast least square deconvolution. Specifically, this algorithm is able to solve a positive definite covariance matrix equations in $O(N^2)$ operations instead of $O(N^3)$ operations. Taking advantage of the inherent structure of the underlying deconvolution problem, this algorithm is guaranteed to be numerically stable since only unitary transformation is used. This algorithm has been implemented on an experimental human vocal tract cross section estimation system and significant speedup has been observed.

I. INTRODUCTION

Consider a discrete time, causal linear system with known impulse response sequence {h(k)}, random input sequence {u(k)}, and random observation noise sequence {n(k)}. Let {z(k)} be the output of this system, then

$$z(k) = \sum_{m=0}^{N-1} h(m)u(k-m) + n(k) \quad (1)$$

for k = 1, 2, ..., N. Denoted by $\mathbf{u} = [u(1)\ u(2), \cdots u(N)]^t$, $\mathbf{z} = [z(1)\ z(2)\ \cdots\ z(N)]^t$, $\mathbf{n} = [n(1)\ n(2), \cdots, n(N)]^t$, and

$$\mathbf{H} = \begin{bmatrix} h(0) & 0 & 0 & \cdots & 0 \\ h(1) & h(0) & 0 & \cdots & . \\ . & h(1) & h(0) & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & 0 \\ h(N-1) & h(N-2) & . & \cdots & h(0) \end{bmatrix}$$

we have

$$\mathbf{z} = \mathbf{H}\,\mathbf{u} + \mathbf{n} \quad (2)$$

Let us further assume that $E\{\mathbf{u}\,\mathbf{n}^*\} = 0$, $E\{\mathbf{u}\,\mathbf{u}^*\} = \nu^2\mathbf{I}_N$, where $\mathbf{I}_N$ is the N by N identity matrix. Thus,

$$E\{\mathbf{z}\,\mathbf{z}^*\} \equiv \Omega = \nu^2\,\mathbf{H}\,\mathbf{H}^t + \sigma^2\,\mathbf{I}_N .$$

Now, the least mean square estimate of the input vector $\mathbf{u}$ given $\mathbf{H}$, $\mathbf{z}$, ν^2, and σ^2 can be found as:

$$\hat{\mathbf{u}} = \nu^2\,\mathbf{H}^t\,\Omega^{-1}\,\mathbf{z} \quad (3)$$

This is a least square *deconvolution* procedure and has been employed in many signal processing applications such as signal recovery, system identification, seismic deconvolution [1] and related areas.

To compute $\hat{\mathbf{u}}$, $O(N^3)$ operations are required to invert the matrix Ω via conventional Cholesky factorization algorithm [2]. However, making use of the (triangular) Toeplitz structure of the $\mathbf{H}$ matrix, a fast algorithm will be devised below which reduces the computation complexity from $O(N^3)$ to $O(N^2)$ operations.

II. A ROTATION BASED ALGORITHM

First, note that the Ω matrix can be factorized into the product of two matrices

$$\Omega = \mathbf{T}^t\,\mathbf{T} = \begin{bmatrix} \nu\,\mathbf{H} & \sigma\,\mathbf{I}_N \end{bmatrix} \begin{bmatrix} \nu\,\mathbf{H}^t \\ \sigma\,\mathbf{I}_N \end{bmatrix} \quad (4)$$

Our strategy is to compute the QR factorization of the Toeplitz matrix $\mathbf{T}$ by nullifying the lower part $\mathbf{I}_N$ matrix via plane rotations. If $\mathbf{T} = \mathbf{Q}\,\mathbf{R}$, where $\mathbf{Q}$ is an orthonormal matrix, and $\mathbf{R}$ is an upper triangular matrix, then the $\mathbf{R}$ matrix will also be the Cholesky upper triangular factor of the Ω matrix. Let us define a 2N by 2N unitary matrix $c(k,\theta) = [c_{mn}]$ such that

$$c_{mn} = \begin{cases} \mathit{Cos}\ \theta & k \le m = n \le 2N-k; \\ \mathit{Sin}\ \theta & k \le m \le N,\ n = m+N-k+1; \\ -\mathit{Sin}\ \theta & k \le n \le N,\ m = n+N-k+1; \\ 1 & m = n;\ m < k,\ \mathit{or}\ m > 2N-k+1; \\ 0 & \mathit{otherwise.} \end{cases}$$

Clearly, choosing $\theta = \tan^{-1}[\sigma/\nu h(0)]$ and compute $\mathbf{T}^{(1)} = c(1,\theta)\,\mathbf{T}$ will nullify all the diagonal elements in the lower half of the $\mathbf{T}$ matrix, Meanwhile, fill-in elements, which also exhibit Toeplitz structure, appear above the nullified diagonal as a result of the Toeplitz structure of the $\nu\,\mathbf{H}^t$ matrix and the

---

*Supported by National Science Foundation under contract No. MIP-8896111.

** Supported by National Institute of Health under grants NS-21516, and NS-16373.

$\sigma\, \mathbf{I}_N$ matrix. Note that since the first column of the $\mathbf{T}^{(1)}$ matrix contains only one (the first) non-zero element, the first row of that matrix must be the first row of the desired **R** matrix. This concludes the first iteration of the fast algorithm. Next, similar to the first iteration, constructing $c(2,\theta')$ and pre-multiplying it to the resulting $\mathbf{T}^{(1)}$ matrix, the first upper sub-diagonal elements of the lower half of the $\mathbf{T}^{(1)}$ matrix can be nullified as well. After N such rotation operations, an upper triangular matrix, which corresponds to the QR factorization of the original **T** matrix emerges in the upper half of the $\mathbf{T}^{(N)}$ matrix. This upper triangular matrix is exactly the desired Cholesky factor of the $\mathbf{\Omega}$ matrix.

Explicit multiplication of $c(k,\theta)\ \mathbf{T}^{(k-1)} = \mathbf{T}^{(k)}$ would take $O(N^2)$ operations per iteration. However, using the fact that the $c(k,\theta)$ matrix operates on two upper triangular Toeplitz matrices, the $\mathbf{T}^{(k)}$ matrix can be determined by performing plane rotations on the k-th row and the $(N+1)^{-th}$ row of the $\mathbf{T}^{(k-1)}$ matrix exclusively. This reduces the per-iteration operation counts to O(N). With N iterations, the total operations for triangular factorization of the $\mathbf{\Omega}$ matrix then is reduced from $O(N^3)$ to $O(N^2)$. Since the remaining operations in (5) such as back/forward substitution, matrix vector multiplications, take only $O(N^2)$ operations, the total number of operations for computing $\hat{\mathbf{u}}$ is $O(N^2)$.

It is not difficult to check that this algorithm is a special case of a more general class fast Toeplitz QR factorization algorithm [3]-[6]. It also has an origin which traces back to the generalized Schur algorithm [7],[8]. One distinct feature of this algorithm is that only plane rotation is required. This guarantees its numerical stability.

III. APPLICATIONS

This fast least square deconvolution algorithm has been incorporated in an experimental system for human vocal tract cross section parameter estimation. The output of the vocal tract in response to an acoustic input follows the linear convolutional model, and the estimation of the reflection impulse response can be posed as a deconvolution of a known input from a noisy measurement of the output. In the apparatus described by Milenkovic [9], the input and output waveform both change in response to changes in the acoustic loading of the measurement system for each different vocal tract shape. We seek to solve the deconvolution problem as efficiently as possible in the case where we cannot precompute the part of the solution derived from the input pulse because the input pulse varies.

In the vocal tract measurement experiment, we would like to compute vocal tract cross section values at .5 cm intervals for a distance of up to 30 cm. This means that 60 is a lower bound on the number of samples in our waveforms to be deconvolved. With this in mind, we have determined computation times using both the conventional Cholesky as well as the new algorithm for N ranging from 64 to 256. The computations were run on a 386-XT Motherboard with an Intel 387 math coprocessor running a 16 MHz. The programs were written in Turbo Pascal Version 4.0 except for the vector dot product operations and the Jabobi's rotations, which were written in assembly language and linked to the main program. The operations are all in 32 bit IEEE floating point format. The results are listed in the table below.

Size (N)	Cholesky	New	Speed-up
64	.77	.22	3.5
128	4.6	.82	5.61
256	32.7	3.25	10.1

REFERENCES

[1] Chi, C.-Y, and J. Mendel, *IEEE Trans. on ASSP*, Vol. 32, No. 6, Dec. 1984, pp. 1145- 1153.

[2] Golub, G.H., and C.F. Van Loan, *Matrix Computations*, John Hopkins Univ. Press, Baltimore, MA, 1983.

[3] Bojanczyk, A.W., R.P. Brent, and F. R. De Hoog, *Numer. Math.*, vol. 49, 1986, pp. 81-94.

[4] Chun, J, T. Kailath, and H. Lev-Ari, *SIAM J. Sci. Stat. Comput.*, vol. 8, No.6, 1987, pp. 899-913.

[5] Cybenko, G., *SIAM J. Sci. Stat. Comput.*, vol. 8, 1987, pp. 734-740.

[6] Sweet, D.R. *Numer. Math.*, vol. 43, 1984, pp. 1-21.

[7] Lev-Ari, H, and T. Kailath, *IEEE Trans. Info. Theory*, Vol.30, 1984, pp. 2-16.

[8] Hu, Y.H., *Proc. ICASSP 88*, New York, NY, April 1988, pp. 1659-1662.

[9] P. Milenkovic, *IEEE Trans. ASSP*, Vol. ASSP-35, pp. 1089-1100, 1987.

RESIDUE COMPLEX MULTIPLIERS

ALEXANDER SKAVANTZOS
Louisiana State University
Baton Rouge, LA

THANOS STOURAITIS
The Ohio State University
Columbus, OH

ABSTRACT

This paper introduces a new complex multiplier which is based on polynomial approximations of complex numbers and their processing via a recently developed parallel arithmetic system. The new multiplier allows a variety of implementation options and exhibits performance up to 3.5 times better than traditional techniques in a multiplicative intensive environment.

A number of important signal processing and communication applications use algorithms which rely heavily on complex multiplications. Such algorithms include the convolution of complex signals, autocorrelation, cross-correlation, as well as computation of the Discrete Fourier Transform (DFT) using Fast Fourier Transform (FFT) techniques.

In this paper an attractive decomposition technique of the complex product $< A \cdot B >_{(2^n-1)}$ is investigated, where $< x >_m$ denotes the operation $x \bmod m$. This technique results in totally parallel multiplier modules. It is based on a recently developed arithmetic system, the Polynomial Residue Number System (PRNS), in which totally parallel polynomial multiplication can be achieved provided that the arithmetic takes place in some carefully chosen ring [1]. The PRNS(N) examines the problem of multiplying two $(N-1)$st-order polynomials mod $x^N \pm 1$ over some modular ring $Z_m = \{0, 1, \ldots, m-1\}$—a ring which is closed with respect to the operations of addition and multiplication mod m. It is a generalization of other number systems like the Quadratic RNS [2,3,4]. By factorizing the polynomials $x^N \pm 1$ in N distinct factors as in

$$x^N \pm 1 \equiv (x - r_0)(x - r_1) \ldots (x - r_{N-1}), \quad r_i \in Z_m, \quad i = 0, \ldots, N-1 \tag{1}$$

and using the PRNS(N) [1] isomorphic mapping, the product of two $(N-1)$st-order polynomials mod $(x^N \pm 1)$ in Z_m can be computed in a totally parallel manner with only N multiplications and no additions at all.

The contribution of the paper is that complex numbers with n-bit real and imaginary parts can be approximated with seventh-order polynomials with $\frac{n}{4}+1$ bits per polynomial coefficient. The parallel PRNS can then perform the polynomial product in a parallel manner and much faster than otherwise. Consider a complex number $A = \Re(A) + j\Im(A)$ with $\Re(A)$ and $\Im(A)$ the n-bit real and imaginary parts of A respectively, where without any loss of generality n is considered to be a multiple of four. By allowing a small approximation error to occur, one can approximate complex numbers using the model

$$\begin{aligned} A &= \Re(A) + j\Im(A) \approx a_0 2^{\frac{3n}{4}} + a_1 2^{\frac{n}{2}} x + a_2 2^{\frac{n}{4}} x^2 + a_3 x^3 \\ &+ a_4 2^{\frac{3n}{4}} x^4 + a_5 2^{\frac{n}{2}} x^5 + a_6 2^{\frac{n}{4}} x^6 + a_7 x^7. \end{aligned} \tag{2}$$

where $x = e^{j\frac{\pi}{8}}$ and the a_is are given by

$$a_i = [\bar{a}_i]_{\frac{n}{4}+1}, \; i = 0, \ldots, 7, \tag{3}$$

where the symbol $[\;]_m$ denotes rounding to m bits. The entries $\bar{a}_i$ have been found to be

$$\begin{array}{ll} \bar{a}_0 = \Re_3(A) & \bar{a}_1 = \Re_2(A)\cos\frac{\pi}{8} + \Im_2(A)\sin\frac{\pi}{8} \\ \bar{a}_2 = \frac{\Re_1(A)+\Im_1(A)}{\sqrt{2}} & \bar{a}_3 = \Re_0(A)\sin\frac{\pi}{8} + \Im_0(A)\cos\frac{\pi}{8} \\ \bar{a}_4 = \Im_3(A) & \bar{a}_5 = -\Re_2(A)\sin\frac{\pi}{8} + \Im_2(A)\cos\frac{\pi}{8} \\ \bar{a}_6 = \frac{\Im_1(A)-\Re_1(A)}{\sqrt{2}} & \bar{a}_7 = -\Re_0(A)\cos\frac{\pi}{8} + \Im_0(A)\sin\frac{\pi}{8}. \end{array} \tag{4}$$

This form allows the n-bit two-tuple $(\Re(A), \Im(A))$ to be approximated by the $\frac{n}{4}$-bit eight tuple $(a_0, a_1, \ldots, a_7)$ used in the multiplication scheme presented next.

Consider two complex numbers $A \approx (a_0, a_1, \ldots, a_7)$ and $B \approx (b_0, b_1, \ldots, b_7)$ and take into account that $x^8 = e^{j\pi} = -1$, $x^i = -x^{<i>_8}$, and $< 2^n >_{(2^n-1)} = 1$. Their product mod $(2^n - 1)$ can be given as

$$\begin{aligned} < A \cdot B >_{(2^n-1)} = \; & < p_0 2^{\frac{n}{2}} + p_1 2^{\frac{n}{4}} x + p_2 x^2 + p_3 2^{\frac{3n}{4}} x^3 + p_4 2^{\frac{n}{2}} x^4 \\ & + p_5 2^{\frac{n}{4}} x^5 + p_6 x^6 + p_7 2^{\frac{3n}{4}} x^7 >_{2^n-1}, \end{aligned} \tag{5}$$

where the coefficients p_i are the coefficients of the polynomial product

$$\begin{aligned} P(x) = \; & < A(x)B(x) >_{(x^8+1)} \\ = \; & < (a_0 + a_1 x + \cdots + a_7 x^7) \cdot (b_0 + b_1 x + \cdots + b_7 x^7) >_{(x^8+1)} \\ = \; & p_0 + p_1 x + \cdots + p_7 x^7. \end{aligned} \tag{6}$$

Choosing a modular ring Z_m so that the polynomial x^8+1 can be factorized in eight distinct first-degree factors in Z_m, the polynomial product of Eq. (6) can efficiently be computed with only eight multiplications (in parallel) using a PRNS(8). The approximation of the complex product $< A \cdot B >_{(2^n-1)}$ requires scaling of the computed coefficients p_i by powers of two and evaluating Eq. (5) for $x = e^{j\frac{\pi}{8}}$.

The proposed complex PRNS multiplier has been architected and compared to traditional ones. The results of this comparison are summarized in Table 1.

n (bits)	t_{trad} (Δ)	t_{PRNS} (Δ)	$\frac{t_{trad}}{t_{PRNS}}$
28	203	212	0.96
60	427	277	1.54
92	651	342	1.90
124	875	406	2.16

Table 1 Propagation delays of the traditional and PRNS-based complex multipliers for one multiplication. Δ is the propagation delay of a NAND gate of the used technology.

It can be observed that the performance of the PRNS(8)-based complex multipliers is increased by a factor greater than two for large word lengths when compared to traditional implementations. The speed advantages of the PRNS(8) multiplier will be amplified if it is operating in a multiplicative intensive environment.

REFERENCES

[1] A. Skavantzos, "The Polynomial Residue Number System and its Applications," Ph.D. Thesis, University of Florida, Gainesville, FL, 1987.

[2] J.V Krogmeier and W.K. Jenkins, "Error Detection and Correction in Quadratic Residue Number Systems," in *Proceedings of 26th Midwest Symposium on Circuits and Systems*, pp. 408–411, August 1983.

[3] S.H. Leung, "Application of Residue Number Systems to Complex Digital Filters," in *Proceedings of Fifteenth Asilomar Conference on Circuits, Systems, and Computers*, pp. 70–74, November 1981.

[4] F.J. Taylor, G. Papadourakis, A. Skavantzos, and A. Stouraitis, "A Radix-4 FFT Using Complex RNS Arithmetic," *IEEE Transactions on Computers*, vol. C-34, no. 6, pp. 573–576, June 1985.

EFFICIENT IMPLEMENTATION OF CHIRP Z-TRANSFORM USING CORDIC ALGORITHM*

Yu Hen Hu

Dept. of Electrical & Computer Engineering
University of Wisconsin - Madison
Madison, WI 53706

S. Naganathan

Electrical Engineering Department
Southern Methodist University
Dallas, TX 75275

ABSTRACT

In this paper, an efficient implementation of the Chirp Z Transform (CZT) using the CORDIC (COordinate Rotation DIgital Computer) algorithm is presented. The proposed implementation is most effective when only a very few complex frequencies on the Z-plane are to be evaluated via CZT. It is shown that the *norm correction (normalization)* operation which is often considered as a computation overhead is converted as an useful scaling operation of the CZT algorithm. Since the desired frequencies of CZT are specified before computation, it is possible to reduce the total number of CORDIC iterations by finding a most economic representation of the angle using a technique called *Angle Recoding*. An algorithm is developed, which is able to reduce the total number of iterations by at least 50 percent without affecting the computational accuracy.

I. INTRODUCTION

In the formulation of several orthogonal transformation based digital signal processing algorithms such as fast (discrete) Fourier transformation, and QR factorization for least square estimations, a major portion of operations involve a rotation of a two by one vector through a certain angle [1]. Such an operation admits an efficient hardware implementation using a rotation based arithmetic algorithm known as CORDIC [2-3]. Recently, a number of research efforts have been made to design VLSI (very large scale integration) based CORDIC processors circuits [4-5] for highly concurrent computation encountered in digital signal processing.

The Chirp Z-Transform [6] evaluates the Z-Transformation of a discrete sequence $\{x(n)\}$ at the points $z_m = A\,W^{-m}$ for $m = 0, 1, \ldots, M\text{-}1$; where $W = W_o\, e^{-j\phi_o}$ and $A = A_o\, e^{j\theta_o}$ with W_o, A_o, ϕ_o, and θ_o being real numbers. That is,

$$X(z_m) = \sum_{n=0}^{N-1} x(n)\, z_m^{-n} = \sum_{n=0}^{N-1} x(n)\, A^{-n}\, W^{mn} \qquad (1)$$

Traditionally, CZT may be evaluated using fast Fourier transformation using $O((N + M - 1)\log_2 (N + M - 1))$ operations [6]. However, when only few chirp frequencies are needed ($M << N$), it would be more efficient to evaluate eq. (1) directly using a recurrence formulation: For $n = 0$ to $N\text{-}1$,

$$B(n) = x(N-1-n) + e^{-j\theta_c} K_c\, B(n-1) \qquad (2)$$

where $B(-1) = 0$, $\theta_c = \theta_o + m\,\phi_o$, and $K_c = A_o^{-1}\, W_o^{m}$. Denoting respectively subscripts "r" and "i" to be the real and imaginary part of a complex number, equation (2) can be expressed as:

$$\begin{bmatrix} B_r(n) \\ B_i(n) \end{bmatrix} = K_c \begin{bmatrix} \cos\theta_c & \sin\theta_c \\ -\sin\theta_c & \cos\theta_c \end{bmatrix} \begin{bmatrix} B_r(n-1) \\ B_i(n-1) \end{bmatrix} + \begin{bmatrix} x_r(N-1-n) \\ x_i(N-1-n) \end{bmatrix} \qquad (3)$$

for $n = 0,1\ldots N\text{-}1$. In eq.(3) a plane rotation is performed on a two by one vector $[B_r(n-1), B_i(n-1)]^t$ through an angle θ_c. The result is then scaled by a factor K_c and then added to the data vector $[x_r(N-1-n), x_i(N-1-n)]^t$. In what follows, the focus will be on the implementation of the plane rotation and scaling operation using the CORDIC algorithm.

II. CORDIC IMPLEMENTATION OF CZT

Let us consider the approximation of the angle θ_c by an algebraic sum of p elementary angles $\{a(i);\ i = 0, p\text{-}1\}$:

$$\theta_c = \sum_{i=0}^{p-1} u(i)\, a(i) + \varepsilon \qquad (4)$$

where ε is an angle approximation error whose magnitude is negligible in practical computation

---

* This research is supported by National Science Foundation under contract numbers DCI-8609283, MIP-8896111.

[3]. If one further chooses $a(i) = \tan^{-1}(2^{-i})$, then the plane rotation and scaling operations in eq. (3) can be computed as:

$$\begin{bmatrix} B'_r(n) \\ B'_i(n) \end{bmatrix} = K_c \prod_{i=0}^{p-1} \cos u(i)a(i) \prod_{i=0}^{p-1} \begin{bmatrix} 1 & u(i)2^{-i} \\ -u(i)2^{-i} & 1 \end{bmatrix} \times \begin{bmatrix} B_r(n-1) \\ B_i(n-1) \end{bmatrix} \tag{5}$$

It is very important to note that the rotation through the angle θ_c is now accomplished by p rotations through p elementary angles using O(p) simple shift-and-add operations. The last expression in eq (5), excluding K_c, in fact, is the CORDIC formulation for computing plane rotations of the vector $[B_r(n-1)\ B_i(n-1)]^t$. The multiplication by $\prod_{i=1}^{p-1}\cos u(i)a(i) = K(p)$ is an intrinsic *norm correction* operation[1] in CORDIC and is often considered a computation overhead. Note that K(p) will be a constant, once $\{u(i)a(i);\ i = 0, p\text{-}1\}$ are determined.

It can easily be seen, that the norm correction factor K(p) should be combined (premultiplied) with the scaling factor K_c to simplify computation. In doing so, we convert an overhead operation in CORDIC, into an useful scaling computation in CZT. Since K_c is a constant for a specific chirp frequency m, the product $K_c\,K(p)$ should also be a constant and can be computed in advance. Therefore, multiplier recoding methods, such as the modified Booth recoding method [7], will be applicable to reduce the required numbers of addition/subtraction operations to no more than one third of the word length.

Since the rotation angle θ_c is known in advance, a technique called *angle recoding* is presented below which facilitates reduction in the number of CORDIC iterations, by letting some of the u(i)s take the value of 0 in addition to ± 1, so that the corresponding CORDIC iteration may be skipped without affecting the result.

The CORDIC Angle Recoding Problem: Given $\{a(i) = \tan^{-1} 2^{-i};\ i = 0, p\text{-}1\}$ *and an angle* θ_c *satisfying* $|\theta_c| < \sum_{i=0}^{p-1} a(i)$, *find* $\{u(i);\ i = 0 \text{ to } p\text{-}1,\ u(i) = 0, \pm 1\}$ *such that* $\sum_{i=0}^{p-1} |u(i)|$ *is minimized.*

An optimum solution to the above problem will take $O(3^p)$ operations which is impractical to evaluate for large value of p. Below, a suboptimal solution which takes only $O(p^2)$ operations is proposed. Without loss generality, it shall be assumed[2] that $|\theta_c| \le a(0)$, and $u(i) = 0$ initially. Then an i_0, $0 \le i_0 \le p-1$ can be chosen such that,

$$\big|\, |\theta_c| - a(i_0) \,\big| = \min_{0 \le i \le p-1} \big|\, |\theta_c| - a(i) \,\big|$$

Define $\delta(1) = |\theta_c| - a(i_0)$, one should set $u(i_0) =$ sign of $\delta(1)$. If $|\delta(1)| \ge a(p\text{-}1)$, above procedure will be repeated to compute $\delta(2)$, $\delta(3)$, and so on, until $|\delta(n^*)| < a(p\text{-}1)$ for some integer n^*. It can be shown that with this method $n^* \le p/2$. Due to space limitation, the proof will not be included.

To summarize, in this paper, it is demonstrated that the CORDIC algorithm is a natural candidate for implementing the CZT. By solving a *CORDIC Angle Recoding problem*, the number of CORDIC iterations can be reduced to speed up the computation without affecting the accuracy.

REFERENCES

[1] K.R. Liu and K.Yao, "On Uniform One-Chip VLSI Design considerations for some Discrete Orthogonal Transforms", *Proc., of ICASSP, New York,*, April 1988, pp. 2136-2139.

[2] J.E. Volder, "The CORDIC Trigonometric computing technique," *IRE Transactions on Electronic Computers* , Vol. EC-8, No.3, pp. 330-334, Sept. 1959.

[3] J.S. Walther, "A Unified algorithm for elementary functions " *Spring Joint Computer Conference* , 1971.

[4] H.M. Ahmed, *"Signal Processing Algorithms and Architectures"*, Ph.D. dissertation, Dept. Electrical Engineering, Stanford University, Stanford, CA, June 1982.

[5] Y.H. Hu, T.Y. Sung, and H.J. Yu, "Doubly pipelined CORDIC array for digital signal processing algorithms," *Proc., Int'l Conf., on Circuit and Systems, San Jose, CA*, May 1986.

[6] A.V. Oppenheim, and R.W. Schafer, *"Digital Signal Processing"*, Prentice-Hall Inc., N.J, 1975.

[7] J.P. Hayes, *"Computer Architecture and Organization"*, McGraw Hill, 1979.

---

[1] The purpose of this norm correction operation is to preserve the norm of the vector $[B_r(n-1),\ B_i(n-1)]^t$ after rotation.

[2] If $|\theta_c| \ge a(0)$, we simply start with $\theta'_c = \theta_c$ modulo $a(0)$.

NEW RESULTS FOR MAGNETIC AND OPTICAL RECORDING CHANNELS

JACK K. WOLF
University of California at San Diego
San Diego, California

ABSTRACT

Coding for both magnetic and optical recording systems employ both error correction codes and modulation codes. The most common form of modulation codes are binary (d,k) codes whereby at least "d" and at most "k" zeros separate adjacent ones in the encoded binary sequence. Several forms of (d,k) codes are discussed. A new variable rate scheme is described whereby the worst case rate is no less than the fixed rate of presently used codes but where the average rate is strictly higher than these codes. Another code is described whereby control information is contained in the (d,k) sequences. Now, the (d,k) encoded binary sequence has predetermined bits in addition to being a mapping of the data sequence. Several methods of combining error correction and modulation codes are discussed. These include (a) a single sequential decoder matched to both the long constraint length convolutional code and a (d,k) code, (b) a single Viterbi decoder matched to a short constraint length convolutional code and a (d,k) code and (c) a single trellis code and decoder that is both an error correction code and a (d,k) code. Finally a method for controlling the polarity of the output pulses in a magnetic recording system is described.

ON THE PATH WEIGHT ENUMERATORS OF CONVOLUTIONAL CODES

CHI-CHAO CHAO, ROBERT J. McELIECE
Department of Electrical Engineering
California Institute of Technology, Pasadena, California

ABSTRACT

In this paper we compute the average distance profile for the ensemble of time-varying convolutional codes, and find that the result gives a surprisingly accurate prediction of the number of fundamental paths at large distance for fixed codes of moderate to large constraint length. We also estimate the average free distance and obtain a Gilbert-like free distance lower bound for time-varying codes. Finally we present a similar analysis for the weight of information bits for fundamental paths.

1. INTRODUCTION

Consider a fixed convolutional code. For each nonnegative integer d, let A_d denote the number of fundamental paths of weight d. The *free distance* of the code is defined to be the least d such that A_d is not zero:

$$d_{\text{free}} = \min\{d \geq 0 : A_d > 0\}.$$

It is well-known that for a given code rate, the larger the free distance, the better the code will perform at *high* signal-to-noise ratios. In this paper, we introduce a quantity, called the *dominant root* of the code, that appears to govern code performance at *low* signal-to-noise ratios:

$$\alpha = \limsup_{d \to \infty} A_d^{1/d}. \tag{1}$$

We conjecture that for a given code rate, the smaller the value of α, the better the code will perform at low signal-to-noise ratios.

This conjecture is based on experimental evidence gathered by Fabrizio Pollara at JPL, and by the following two theoretical arguments. First, at low SNR one expects the channel noise to be severe, and hence fundamental paths at large distance from the transmitted codepath to be the ones in close competition with it. According to the definition (1), however, the smaller the the value of α, the fewer such paths will exist, and hence the decoder should have a better chance of choosing the correct path. Second, if the code's *path weight enumerator* [1] is defined as

$$A(x) = \sum_{d \geq 0} A_d x^d,$$

then the first-event error probability P_E is bounded above by $A(\gamma)$, where γ is the channel's Bhattacharyya parameter [2]. The largest positive value of γ for which this bound converges is the radius of convergence of $A(x)$, which by a well-known theorem [3, p. 213] is α^{-1}, where α is defined in (1). Thus if code 1 has dominant root α_1 and code 2 has dominant root α_2 with $\alpha_1 < \alpha_2$, then for $\alpha_2^{-1} < \gamma < \alpha_1^{-1}$ the bound for P_E for code 2 diverges but that for code 1 converges, thus again suggesting that the code with smaller value of α will perform better at low SNR.

In principle, the path weight enumerator $A(x)$ can be computed by applying Mason's gain rule, or some other standard combinatorial technique, to the code's labelled state diagram [1] [4]. In general, $A(x)$ is a rational function, with integer coefficients:

$$A(x) = \frac{N(x)}{D(x)}.$$

Since the coefficients A_d of $A(x)$ are nonnegative, it follows from [3, Theorem 7.21] that $x = \alpha^{-1}$ is a singularity of $A(x)$, i.e. that $D(x^{-1}) = 0$. Thus α is the reciprocal of the least-magnitude root of the equation $D(x) = 0$.

This work was supported by the Air Force Office of Scientific Research under Grant No. AFOSR–88–0247.

In this paper we will investigate the dominant roots of various convolutional codes. In Table I we actually compute the α's for some $(2,1)$ convolutional codes of constraint length K from 4 to 12. We compute α as follows (except that for $K = 4$ case α is computed directly from the denominator of $A(z)$). From [5], we know that the maximum number of consecutive all-zero branches that can occur in a nonzero fundamental path is $K-2$ for a rate $1/2$ code of constraint length K. Hence the fundamental path of weight d is at most of length $(K-1)d$. We then compute A_d from the trellis diagram up to some large enough d for a given code and found α approximately by either A_d/A_{d-1} or $\sqrt{A_d/A_{d-2}}$ for some large d. (For some codes $A_d = 0$ for odd d, so we compute $\sqrt{A_d/A_{d-2}}$ for large even d instead of A_d/A_{d-1}.) It is interesting to note that for fixed K all α's are close together and they seem to approach a limit for very large K.

In this paper we explain this interesting phenomenon by considering the ensemble of fixed convolutional codes as a subset of the ensemble of time-varying convolutional codes, and then finding the average number of fundamental paths of each weight for a random time-varying convolutional code. Hence the behavior of the average number of fundamental paths at large distance can be analyzed. The corresponding generating function is also found; from this we can estimate the average free distance of a random time-varying convolutional code. Furthermore, we can obtain a Gilbert-like lower bound on free distance for time-varying codes. Finally we give a similar random analysis for the total weight of information bits for fundamental paths at each distance.

2. AVERAGE DISTANCE STRUCTURES OF RANDOM CONVOLUTIONAL CODES

In this section we will compute the average distance profile for the ensemble of time-varying convolutional codes. A time-varying convolutional code [2], [6] is a convolutional code of which the generator polynomials may be changed after each time unit. In other words, the tap positions of modulo-2 adders in the shift register encoder is reselected after each shift of the bits. Now let's consider the ensemble of all time-varying convolutional codes which include the ensemble of fixed convolutional codes as a subset. A uniform probability measure is further imposed to each code by randomly reselecting the tap positions after each shift. This can be done by hypothetically flipping a fair coin for each tap position. Then the output will be a random binary vector after each shift.

Let $\overline{A_d}$ denote the average number of fundamental paths of weight d in the ensemble of all (n,k) time-varying convolutional codes of constraint length K. For simplicity, only the codes with $2^{k(K-1)}$ states in their corresponding state diagram will be considered thereafter. Equivalently , we consider only the (n,k) codes whose shift-register encoders have k shift registers all of the same length K. We have

$$\overline{A_d} = \sum_l \overline{A_{d,l}}, \tag{2}$$

where $A_{d,l}$ is the number of fundamental paths of weight d and length l. Under the uniform probability measure, each branch of a path is a random n-dimensional binary vector, and each nl-dimensional binary vector has the same probability to be a fundamental path of length l. Let T_l denote the number of fundamental paths of lenth l for any (n,k) convolutional codes of constraint length K. Then the probability of being a fundamental path for any nl-dimensional binary vector is $T_l/2^{nl}$. Since there are total $\binom{nl}{d}$ nl-dimensional binary vector of weight d, $\overline{A_{d,l}}$ can be expressed as

$$\overline{A_{d,l}} = \binom{nl}{d}\frac{T_l}{2^{nl}}. \tag{3}$$

We show in the appendix that the generating function $T(y)$ of T_l for any (n,k) code being considered is given by

$$T(y) = \frac{P(y)}{Q(y)} = \frac{(2^k-1)y^K(1-y)}{1-2^k y + (2^k-1)y^K}. \tag{4}$$

For $K > 2$, the denominator $Q(y)$ and its derivative can be shown to be relatively prime. Hence $Q(y)$ is squarefree. For $K > 2$, Equation (4) can therefore be partial-fraction expanded to

$$T(y) = 1 - y + \sum_{i=1}^{K-1} \frac{c_i}{1-r_i y},$$

where $r_i, i = 1, \ldots, K-1$, are the reciprocals of the roots of $1 - 2^k y + (2^k - 1)y^K$ other than 1 and

$$c_i = \frac{-r_i P(r_i^{-1})}{Q'(r_i^{-1})}.$$

Hence we obtain

$$T_l = \phi_l + \sum_{i=1}^{K-1} c_i r_i^l, \tag{5}$$

where

$$\phi_l = \begin{cases} 1, & \text{if } l = 0, \\ -1, & \text{if } l = 1, \\ 0, & \text{if } l \geq 2. \end{cases}$$

From (2), (3) and (5), we will show that for $K > 2$,

$$\overline{A_d} = \theta_d + \frac{1}{n} \sum_{i=1}^{K-1} \sum_{\delta:\delta^n = r_i} \frac{c_i}{1 - \frac{\delta}{2}} \left(\frac{\delta}{2 - \delta} \right)^d, \tag{6}$$

where

$$\theta_d = \begin{cases} 1 - \frac{1}{2^n}, & \text{if } d = 0, \\ \frac{-\binom{n}{d}}{2^n}, & \text{if } 1 \leq d \leq n, \\ 0, & \text{if } d > n. \end{cases}$$

The proof goes as follows. For $d > n$, from (2), (3) and (5) we can obtain

$$\begin{aligned} \overline{A_d} &= \sum_{i=1}^{K-1} \sum_{l=\lceil \frac{d}{n} \rceil}^{\infty} \binom{nl}{d} \frac{c_i r_i^l}{2^{nl}} \\ &= \sum_{i=1}^{K-1} c_i \sum_{\substack{j=nl \\ j \geq d}} \frac{j(j-1)\cdots(j-d+1)}{d!} \left(\frac{r_i^{\frac{1}{n}}}{2} \right)^j, \end{aligned}$$

where $r_i^{1/n}$ can be taken as any n-th root of r_i. In order to compute the above series, we define the following function. Let $f(t)$ be $1 + t^n + t^{2n} + \cdots + t^{ln} + \cdots$; then the d-th derivative of $f(t)$ can be found to be

$$f^{[d]}(t) = t^{-d} \sum_{\substack{j=nl \\ j \geq d}} j(j-1)\cdots(j-d+1) t^j.$$

It thus follows that

$$\overline{A_d} = \sum_{i=1}^{K-1} c_i \cdot \frac{t^d}{d!} f^{[d]}(t) \bigg|_{t = \frac{r_i^{\frac{1}{n}}}{2}}, \quad \text{for } d > n. \tag{7}$$

It is not difficult to see that

$$f(t) = \frac{1}{1 - t^n} = \frac{1}{n} \sum_{s=0}^{n-1} \frac{1}{1 - \omega^s t},$$

where $\omega = e^{i \frac{2\pi}{n}}$. By this form the d-th derivative of $f(t)$ is easy to compute and we can have

$$f^{[d]}(t) = \frac{1}{n} \sum_{s=0}^{n-1} d! \cdot \omega^{sd} \cdot (1 - \omega^s t)^{-(d+1)}. \tag{8}$$

Combining (7) and (8), we get

$$\overline{A_d} = \frac{1}{n} \sum_{i=1}^{K-1} \sum_{s=0}^{n-1} \frac{c_i}{1 - \omega^s \cdot \frac{r_i^{\frac{1}{n}}}{2}} \left(\frac{\omega^s r_i^{\frac{1}{n}}}{2 - \omega^s r_i^{\frac{1}{n}}} \right)^d, \quad \text{for } d > n.$$

Since $\omega^s r_i^{1/n}$, $s = 0, 1, \ldots, n-1$, contain all the n-th roots of r_i, $\overline{A_d}$ can finally be expressed as in (6) for $d > n$. The proofs for $d = 0$ and $1 \leq d \leq n$ are the same as the above except that a few modifications for the offset θ_d are needed.

Now we want to investigate the behavior of $\overline{A_d}$ when d is large. Let r be the reciprocal of the least-magnitude root of $1 - 2^k y + (2^k - 1)y^K$ which is the the denominator of $T(y)$ and δ be any n-th root of any r_i. From [3, Theorem 7.21] the existence of a real positive r is clear because $T(y)$ is a nonnegative series. Since

$$\left|\frac{\delta}{2-\delta}\right| \leq \frac{|\delta|}{||2| - |\delta||} \leq \frac{\sqrt[n]{r}}{2 - \sqrt[n]{r}},$$

it follows by (6) that for large d, $\overline{A_d}$ satisfies

$$\overline{A_d} \approx \beta \hat{\alpha}^d,$$

where β is some constant independent of d and

$$\hat{\alpha} = \frac{\sqrt[n]{r}}{2 - \sqrt[n]{r}}.$$

For large K, r is very close to 2^k, and then $\hat{\alpha}$ is very close to $2^R/(2 - 2^R)$, where $R = k/n$ is the rate.

We compute $\hat{\alpha}$ for $(2,1)$ codes of constraint lengths from 3 to 12 in Table II. Comparing Table I and II, we can see that for fixed K, α's in Table I are quite close to $\hat{\alpha}$ in Table II, especially for large K. Some kind of law of large numbers appear to be in operation. If we pick a convolutional code randomly, then its α is expected to be close to the average $\hat{\alpha}$. Note that for $(2,1)$ codes $\hat{\alpha}$ will approach $\sqrt{2}/(2-\sqrt{2}) = 1 + \sqrt{2}$, which is 2.4142136..., for large constraint length. All the above results can facilitate predicting A_d at large distance for codes of moderate to large constraint length. Also note that the α's for the codes usually used in practice, of which the encoders have both ends of shift registers tapped to the modulo-2 adders, are closer to the average $\hat{\alpha}$'s than those for systematic codes (the last row of each constraint length in Table I).

3. FREE DISTANCES

In this section we first compute the generating function $\overline{A}(x)$ for $\overline{A_d}$; then we can estimate the average free distance and obtain a Gilbert-like lower bound on free distance for time-varying convolutional codes. From (2) and (3) we can have

$$\overline{A}(x) = \sum_{d=0}^{\infty} \overline{A_d} x^d = \sum_{d=0}^{\infty} \sum_{l=\lceil \frac{d}{n} \rceil}^{\infty} \binom{nl}{d} \frac{T_l}{2^{nl}} x^d.$$

Change the order of summations, and we can obtain

$$\overline{A}(x) = \sum_{l=0}^{\infty} \frac{T_l}{2^{nl}} \sum_{d=0}^{nl} \binom{nl}{d} x^d = \sum_{l=0}^{\infty} \frac{T_l}{2^{nl}} (1+x)^{nl}.$$

Therefore $\overline{A}(x)$ can be found to be

$$\begin{aligned} \overline{A}(x) &= T\left(\left(\frac{1+x}{2}\right)^n\right) \\ &= \frac{(2^k-1)(1+x)^{nK}[2^n - (1+x)^n]}{2^{n(K+1)} - 2^{nK+k}(1+x)^n + 2^n(2^k-1)(1+x)^{nK}}. \end{aligned} \tag{9}$$

If we define $\hat{d}_{\text{free}}$ to be

$$\hat{d}_{\text{free}} = \min\left\{d : \overline{A_d} \geq 1\right\},$$

then intuitively $\hat{d}_{\text{free}}$ constitutes an estimate of the average free distance of a random time-varying convolutional code. Furthermore, let's define

$$\tilde{d}_{\text{free}} = \min\left\{d : \sum_{i=0}^{d} \overline{A_i} \geq 1\right\}.$$

Then $\tilde{d}_{\text{free}}$ is a Gilbert-like lower bound on free distance for time-varying convolutional codes. This lower bound guarantees the existence of at least one time-varying code such that the free distance is bounded below. It is so because $\sum_{i=0}^{d} A_i$ must be greater than or equal to 1 for any time-varying codes of given rate and constraint length if d is the corresponding maximal free distance. It becomes clear to say that this lower bound is Gilbert-like when we consider its analogy for block codes. For random (n,k) block codes, $\overline{A_d}$ becomes $\binom{n}{d}/2^{n-k}$ if $\overline{A_d}$ is the average number of codewords of weight d. Then $\sum_{i=0}^{d} \overline{A_i} \geq 1$ is just the Gilbert-Varshamov bound for block codes except that the upper limit of the summation should be $d-1$ for the bound. It is easy to see that $\tilde{d}_{\text{free}} \leq \hat{d}_{\text{free}}$. Both $\tilde{d}_{\text{free}}$ and $\hat{d}_{\text{free}}$ for codes of given rate and constraint length can be computed numerically from equation (9). In Table III we compute $\tilde{d}_{\text{free}}$ and $\hat{d}_{\text{free}}$ for rate 1/2, 1/3 and 1/4 convolutional codes, and the maximal free distances of corresponding fixed codes are also included for comparison.

Costello [7] obtained a Gilbert-type asymptotic lower bound on free distance for nonsystematic time-varying codes. We restate it here as follows.

$$\lim_{K\to\infty} \frac{d_{\text{free}}}{nK} \geq \frac{R(1-2^{R-1})}{H(2^{R-1})+R-1},$$

where $H(x)$ is the binary entropy function. Compared with Costello's bound, our bound has the advantage that the lower bound on free distance can be found for each finite constraint length while it was infeasible in [7]. Although the asymptotic form of our bound is difficult to get, numerical results show that for $(n,1)$ time-varying codes our lower bound is almost the same as Costello's bound asymptotically. While for (n,k) codes with $k>1$, our bound is worse than Costello's bound asymptotically. One possible explanation is that only the (n,k) codes with $2^{k(K-1)}$ states in their corresponding state diagram are considered in our results.

4. AVERAGE WEIGHTS OF INFORMATION BITS FOR FUNDAMENTAL PATHS

In this section we give a similar random analysis for the total Hamming weight of information bits for fundamental paths at each distance. The *complete path enumerator* [1] is defined by

$$A(x,y,z) = \sum_d \sum_l \sum_i A_{d,l,i} x^d y^l z^i,$$

where the y terms describe the length of fundamental paths ,the z terms describe the Hamming weights of the corresponding inputs, and $A_{d,l,i}$ is the number of fundamental paths of distance d, length l, and input weight i. We can compute $A(x,y,z)$ by the same method in obtaining $A(x)$. In general $A(x,y,z)$ is a rational function, with integer coefficients:

$$A(x,y,z) = \frac{N(x,y,z)}{D(x,y,z)}.$$

If we define $B_d = \sum_l \sum_i i A_{d,l,i}$, then B_d is the total Hamming weight of information bits for fundamental paths at distance d. We can hence obtain the generating function $B(x)$ for B_d by

$$\begin{aligned} B(x) &= \left.\frac{\partial A(x,y,z)}{\partial z}\right|_{y=z=1} \\ &= \frac{N_z(x,1,1)D(x,1,1) - N(x,1,1)D_z(x,1,1)}{D^2(x,1,1)} \\ &= \frac{N_z(x,1,1)D(x) - N(x)D_z(x,1,1)}{D^2(x)}, \end{aligned} \tag{10}$$

where $N(x)$ and $D(x)$ are the numerator and the denominator of $A(x)$, respectively. The quantity B_d is of interest to us because for maximum-likelihood decoding the bit error probability can be bounded above by

$$P_b \leq \frac{1}{k} \left.\frac{\partial A(x,y,z)}{\partial z}\right|_{x=\gamma, y=z=1} = \frac{1}{k} B(x)\Big|_{x=\gamma},$$

where γ is again the channel's Bhattacharyya parameter. Please note that the bound diverges above the same α^{-1} as that for the first-event error probability.

Now we want to find the average $\overline{B_d}$ in the ensemble of all (n,k) time-varying convolutional codes of constraint length K. Let $T_{l,i}$ denote the number of fundamental paths of length l and input weight i. Then under the uniform probability measure, we can have

$$\overline{A_{d,l,i}} = \binom{nl}{d} \frac{T_{l,i}}{2^{nl}}.$$

It hence follows that

$$\begin{aligned} \overline{B_d} &= \sum_{l=0}^{\infty} \sum_{i=0}^{\infty} i \overline{A_{d,l,i}} \\ &= \sum_{l=\lceil \frac{d}{n} \rceil}^{\infty} \frac{\binom{nl}{d}}{2^{nl}} \sum_{i=0}^{\infty} i T_{l,i}. \end{aligned} \tag{11}$$

The generating function $T(y,z)$ for $T_{l,i}$ is found in the appendix. We define $S_l = \sum_i i T_{l,i}$, and then its generating function is given by

$$S(y) = \left. \frac{\partial T(y,z)}{\partial z} \right|_{z=1} = \frac{k 2^{k-1} y^K (1-y)^2}{\left(1 - 2^k y + (2^k - 1) y^K\right)^2}.$$

Similar to Section 2, $S(y)$ can be partial-fraction expanded to

$$S(y) = \sum_{i=1}^{K-1} \left(\frac{e_i}{1 - r_i y} + \frac{f_i}{(1 - r_i y)^2} \right),$$

where r_i, $i = 1, 2, \ldots, K-1$, are the reciprocals of the roots of $1 - 2^k y + (2^k - 1) y^K$ other than 1 and

$$e_i = -\frac{1}{r_i} \frac{d}{dy} \left[S(y)(1 - r_i y) \right] \Big|_{y = r_i^{-1}},$$

$$f_i = S(y)(1 - r_i y)^2 \big|_{y = r_i^{-1}}.$$

We can thus obtain

$$S_l = \sum_{i=1}^{K-1} \left(e_i r_i^l + f_i (l+1) r_i^l \right). \tag{12}$$

Substituting (12) back to (11) and using the same technique in Section 2, we can finally obtain

$$\overline{B_d} = \frac{1}{n} \sum_{i=1}^{K-1} \sum_{\delta : \delta^n = r_i} \left\{ \frac{e_i + f_i}{1 - \frac{\delta}{2}} \left(\frac{\delta}{2 - \delta} \right)^d + \frac{1}{n} \left[\frac{f_i}{1 - \frac{\delta}{2}} \left(\frac{\delta}{2 - \delta} \right)^{d+1} + \frac{f_i}{\left(1 - \frac{\delta}{2}\right)^2} \cdot d \left(\frac{\delta}{2 - \delta} \right)^d \right] \right\}.$$

Hence for large d,

$$\overline{B_d} \approx \beta_1 \left(\frac{\sqrt[n]{r}}{2 - \sqrt[n]{r}} \right)^d + \beta_2 \cdot d \left(\frac{\sqrt[n]{r}}{2 - \sqrt[n]{r}} \right)^d,$$

where r is the reciprocal of the least-magnitude root of $1 - 2^k y + (2^k - 1) y^K$. This result can be expected from the expression of (10). Similar to $\overline{A}(x)$, the generating function $\overline{B}(x)$ for B_d can be found to be

$$\begin{aligned} \overline{B}(x) &= \sum_{l=0}^{\infty} \sum_{d=0}^{nl} \frac{\binom{nl}{d}}{2^{nl}} S_l x^d = \sum_{l=0}^{\infty} \frac{S_l}{2^{nl}} (1+x)^{nl} \\ &= S\left(\left(\frac{1+x}{2} \right)^n \right) = \frac{k 2^{k-1} 2^{nK} (1+x)^{nK} \left[2^n - (1+x)^n \right]^2}{\left[2^{n(K+1)} - 2^{nK+k} (1+x)^n + 2^n (2^k - 1)(1+x)^{nK} \right]^2}. \end{aligned}$$

APPENDIX

In this appendix we will derive $T(y)$ given in (4) by extending the argument in [2, Sec. 4.6], where $T(y)$ for any $(n,1)$ code is found. Actually we will find $T(y,z)$ first, where the z terms describe the Hamming weights of the corresponding input bits. Consider any (n,k) convolutional code of constraint length K with $2^{k(K-1)}$ states in the corresponding state diagram. In other words, we consider only the (n,k) codes whose shift-register encoders have k shift registers all of the same length K. Then $T(y,z)$ is independent of n and the generator polynomials. It is a function only of k and the constraint length K. Since we will use recursions on K to get the expression for $T(y,z)$, we change the notation to $T_K(y,z)$ to emphasize the constraint length. We claim that the following recursion holds which is the same as [4, (4.6.2)]:

$$T_K(y,z) = y \cdot T_{K-1}(y,z) + T_{K-1}(y,z)T_K(y,z), \quad K \geq 2. \tag{13}$$

The reasoning goes as follows. Consider all states prior to the all-zero state in the state diagram. There are $2^k - 1$ of them. Now think all paths reaching any of these next-to-all-zero states as being absorbed. Then the generating function enumerating those paths is just $T_{K-1}(y,z)$ because we could ignore the initial inputs in all shift registers as if we had a code of constraint length $K-1$. If the following input is all-zero, then we get a fundamental path and this case constitutes the first term of (13). If the following input is anything other than all-zero, then we are in the same situation as we are just from the all-zero state. And $T_K(y,z)$ enumerates all those paths. Hence we can obtain the second term of (13).

The initial condition for (13) is

$$T_1(y,z) = \sum_{i=1}^{k} \binom{k}{i} yz^i = \left[(1+z)^k - 1\right] y$$

because any input other than all-zero produces a fundamental path of length 1 for a code of constraint length 1. Then from (13) by induction we can easily show

$$T_K(y,z) = \frac{\left[(1+z)^k - 1\right] y^K (1-y)}{1 - y\left\{1 + \left[(1+z)^k - 1\right](1 - y^{K-1})\right\}}.$$

We can therefore obtain the expression for $T_K(y)$:

$$T_K(y) = T_K(y,z)\big|_{z=1} = \frac{(2^k-1)y^K(1-y)}{1 - 2^k y + (2^k-1)y^K},$$

which reduces to [4, (4.6.6)] for $k = 1$.

REFERENCES

[1] R. J. McEliece, *The Theory of Information and Coding.* Reading, MA: Addison-Wesley, 1977.

[2] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding.* New York: McGraw-Hill, 1979.

[3] E. C. Titchmarsh, *The Theory of Functions.* London: Oxford University Press, 1939.

[4] S. Lin and D. J. Costello, Jr., *Error-Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

[5] G. D. Forney, Jr., "Structural analysis of convolutional codes via dual codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 512–518, July 1973.

[6] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.

[7] D. J. Costello, Jr., "Free distance bounds for convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 356–365, May 1974.

TABLE I
α FOR SOME (2, 1) CONVOLUTIONAL CODES

	Generator polynomials			
K	g_1	g_2	d_{free}	α
4	1101	1111	6	2.20557
4	1011	1101	6	2.21750
4	1001	1011	5	2.20557
4	1110	1101	6	2.08931
4	1010	1101	5	2.07766
4	1111	0111	4	2.06709
4	1001	1000	3	1.93013
5	10011	11101	7	2.31266
5	10101	11101	6	2.29655
5	10111	11111	6	2.30147
5	11101	11001	7	2.29503
5	10001	11111	6	2.34632
5	11111	01111	4	2.25692
5	10101	10000	4	2.09504
6	110101	101111	8	2.35695
6	110001	101001	6	2.35830
6	100101	111111	8	2.36326
6	110011	111011	7	2.35927
6	100011	100111	7	2.34530
6	101111	011001	8	2.35089
6	110101	100000	5	2.18396
7	1011011	1111001	10	2.38762
7	1100101	1000101	7	2.37357
7	1001001	1110011	8	2.38751
7	1100111	1111111	6	2.38699
7	1001111	1100011	8	2.38929
7	1110011	0101001	8	2.38219
7	1111001	1000000	6	2.24920
8	11100101	10011111	10	2.40007
8	11011011	10011011	7	2.39711
8	10011101	11010011	8	2.39768
8	10001001	10101101	8	2.39200
8	10000101	10111001	8	2.40034
8	10111001	01110001	9	2.38478
8	11101111	10000000	6	2.32136
9	101110001	111101011	12	2.40733
9	100100101	100100001	7	2.41385
9	110000001	110101101	9	2.40651
9	101011001	110100111	11	2.40672
9	111100011	101001101	11	2.40582
9	101111011	011001111	10	2.40383
9	111010111	100000000	8	2.34476

TABLE I

(*Continued*)

K	g_1	g_2	d_{free}	α
10	1001110111	1101100101	12	2.41046
10	1011101001	1001010011	9	2.40925
10	1010011001	1001111011	12	2.41026
10	1101101001	1011110101	12	2.41019
10	1111100011	1000011011	10	2.40888
10	1100100011	0111010001	10	2.40999
10	1110111001	1000000000	8	2.35743
11	10011011101	11110110001	14	2.41212
11	11100110111	10001101111	11	2.41125
11	11010000001	10010011111	11	2.41247
11	11000010011	11001001101	11	2.41172
11	11001010101	10000101011	11	2.41251
11	11110101001	01101100001	12	2.40988
11	11110110001	10000000000	8	2.36820
12	100011011101	101111010011	15	2.414
12	111110110111	111101011011	11	2.411
12	101001001101	111000100111	12	2.410
12	110110111001	100001100101	13	2.413
12	101011011001	100100001001	10	2.409
12	101100000101	010100010111	11	2.412
12	110001100001	100000000000	6	2.408

TABLE II

$\hat{\alpha}$ FOR (2, 1) CONVOLUTIONAL CODES

K	$\hat{\alpha}$
3	1.7473273
4	2.1065695
5	2.2699329
6	2.3451684
7	2.3806466
8	2.3977176
9	2.4060505
10	2.4101566
11	2.4121921
12	2.4132048

TABLE III

$\tilde{d}_{\text{free}}$, $\hat{d}_{\text{free}}$ AND $\max d_{\text{free}}$ FOR SOME CONVOLUTIONAL CODES

(a) $R = 1/2$

K	$\tilde{d}_{\text{free}}$	$\hat{d}_{\text{free}}$	$\max d_{\text{free}}$
3	3	3	5
4	4	4	6
5	4	5	7
6	5	5	8
7	6	6	10
8	7	7	10
9	7	8	12
10	8	9	12
11	9	9	14
12	10	10	15
13	10	11	16
14	11	12	16
15	12	13	18
16	13	13	19
17	14	14	20

(b) $R = 1/3$

K	$\tilde{d}_{\text{free}}$	$\hat{d}_{\text{free}}$	$\max d_{\text{free}}$
3	5	6	8
4	6	7	10
5	7	8	12
6	9	10	13
7	10	11	15
8	11	12	16
9	12	14	18
10	14	15	20
11	15	16	22
12	16	18	24
13	18	19	24
14	19	20	26

(c) $R = 1/4$

K	$\tilde{d}_{\text{free}}$	$\hat{d}_{\text{free}}$	$\max d_{\text{free}}$
3	6	10	10
4	8	11	13
5	10	13	16
6	12	14	18
7	14	16	20
8	16	18	22
9	17	20	24
10	19	22	27
11	21	23	29
12	23	25	32
13	25	27	33
14	26	29	36

AN ARBITRARILY TIGHT UNION BOUND FOR CONVOLUTIONAL CODES THROUGH BIFURCATION

Marat V. Burnashev,
Institute for Problems in Information Transmission
USSR Academy of Sciences
Ermolovoy str. 19, 101447, Moscow

Arvind S. Arora, David L. Cohn
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

I. INTRODUCTION

In 1971, Viterbi [1] formulated the Union bound on the first event error probability (FEP) and bit error probability (BEP) of convolutional codes as an infinite series. Each term in the series can be upper bounded using the Chernoff bound. Van de Meeberg [2] and Zigangirov et. al. [3] tightened the term bound of Viterbi. This resulted in the transfer function bound, which makes numerical computation of the Union bound feasible. In 1980, Post [4] presented an analytical formulation for the Union bound by using a generating function for P_k. The paper illustrated the technique with an example for the (2,1,2) code, but the approach loses tractability except for very short codes. None of these approaches addresses the tightness of the bounds. It also turns out that the usefulness of the transfer function bound (as an estimate of the Union bound) is limited to very small values of the cross-over probability p. This paper proposes a new approach, **bounding through bifurcation**, to estimate the performance of convolutional codes, as precisely as desired. Union bounds on first event error and bit error probabilities for rate 1/n convolutional codes on a binary symmetric channel (BSC) are developed.

II. UNION BOUNDS THROUGH BIFURCATION

The transfer function bound was formulated to estimate the Union bound [1,2,3]. For FEP and BEP,

$$P_{Fe} \le P_{FU} \equiv \sum_{k=d_{free}}^{\infty} A_k P_k \le \frac{1}{2}\Gamma_{d_{free}} \left[(1+D)\, T_F(D) + (1-D)\, T_F(-D)\right]\Big|_{D=D_0}, \quad \Gamma_{2k-1} = \Gamma_{2k} \equiv \binom{2k-1}{k} 2^{-2k} \frac{1}{1-2p} \tag{1}$$

$$P_{Be} \le P_{BU} \equiv \sum_{k=d_{free}}^{\infty} B_k P_k \le \frac{1}{2}\Gamma_{d_{free}} \left[(1+D)\, T_B(D) + (1-D)\, T_B(-D)\right]\Big|_{D=D_0}, \quad T_B(D) \equiv \frac{\partial}{\partial N} T(D,N)\Big|_{N=1} \tag{2}$$

Here, A_k and B_k are the multiplicities of k-weight code-paths and the corresponding total information bits. P_k is the error probability due to decoding a k-weight code-path. T_F is the weight enumerating transfer function, evaluated at $D_0 = 4pq$.(Subscripts F and B are used to associate with FEP or BEP, and ˇ and ˆ are used to indicate lower and upper estimates of a variable.)

Bounding through bifurcation can heuristically be described as follows. An infinite number of random sequences contribute to each performance criterion. A set of sequences with the more significant contribution, called the *front-end*, is identified. Its contribution is evaluated exactly, or at least carefully. The contribution of the rest of the sequences, called the *tail*, can be estimated by previously published methods, or by a new approach, such as discussed in the next section The performance estimate can made arbitrarily precise by increasing the number of sequences in the front-end. The Union bound through bifurcation is formulated as follows.

$$P_{FU} \le \sum_{k=d_{free}}^{L} A_k\, P_k + \frac{1}{2}\left[(1+D)\, T_{F,L}(D) + (1-D)\, T_{F,L}(-D)\right]\Big|_{D=D_0}, \quad T_{F,L}(D) = \Gamma_L \left[T_F(D) - \sum_{k=d_{free}}^{L} A_k D^k \right] \tag{3}$$

$$P_{BU} \le \sum_{k=d_{free}}^{L} B_k\, P_k + \frac{1}{2}\left[(1+D)\, T_{B,L}(D) + (1-D)\, T_{B,L}(-D)\right]\Big|_{D=D_0}, \quad T_{B,L}(D) = \Gamma_L \left[T_B(D) - \sum_{k=d_{free}}^{L} B_k D^k \right] \tag{4}$$

The last term in the front-end always has even weight, L. Table 1 compares the Union bound and TFB for a (2,1,3) code, [5, p.330]. The TFB estimate becomes quite loose rapidly; while with bifurcation, a fixed accuracy, 1%, is maintained.

p	TF Bound	.UB BEP	L	% err. in UB	% err. in TFB
0.01	0.333^{-4}	0.311^{-4}	12	0.5%	7.2
0.02	0.479^{-3}	0.412^{-3}	18	0.5%	16.3
0.03	0.341^{-2}	0.262^{-2}	26	0.7%	29.9
0.04	0.242^{-1}	0.156^{-1}	46	0.8%	55.9
0.05	0.532^{-0}	0.214^{-0}	98	6.7%	148.3

(a) BEP

p	TF Bound	UB FEP	L	% err. in UB	% err. in TFB
0.01	0.137^{-4}	0.131^{-4}	10	0.8%	4.6%
0.02	0.156^{-3}	0.141^{-3}	14	0.7%	10.2%
0.03	0.801^{-3}	0.680^{-3}	20	0.8%	18.0%
0.04	0.345^{-2}	0.261^{-2}	34	0.8%	32.1%
0.05	0.235^{-1}	0.130^{-1}	98	1.1%	81.3%

(b) FEP

Table 1: Comparison between Union bound and the transfer function bound for the (2,1,3) code, [5, p.330].
g = (6, 7)

III. FIRST ORDER BOUND ON THE TAIL

Tail estimation requires numerical computation of the transfer function, which is difficult except for a few short codes. The tail can be bounded by a first order function based on the dominant pole $1/r_0$ of the transfer function. Instead of the terms P_k, the ratios A_{L+1}/A_L and P_{L+2}/P_L, denoted Δ_L and $\gamma_L{}^2$, are upper and lower bounded. This leads to first order bounds, $T_{F,L}$ and $T_{B,L}$, for the tails for FEP and BEP in equations (3) and (4).

$$\frac{L+1}{L+2}4pq \le \gamma_{L-1}^2 = \gamma_L^2 \le 4pq \tag{5}$$

$$\frac{1-O(\rho^{\kappa+1})}{1+O(\rho^{\kappa})}r_0 \le \Delta_{F,L} \equiv \frac{A_{L+1}}{A_L} \le \frac{1+O(\rho^{\kappa+1})}{1-O(\rho^{\kappa})}r_0 \,, \quad \rho = |r_1|/|r_0| < 1 \quad , \quad \kappa = L-d_{free} \tag{6}$$

$$\frac{1-O(\rho^{\kappa+1})}{1+O(\rho^{\kappa})}r_0 \le \Delta_{B,L} \equiv \frac{\kappa+1}{\kappa+2}\frac{B_{L+1}}{B_L} \le \frac{1+O(\rho^{\kappa+1})}{1-O(\rho^{\kappa})}r_0 \tag{7}$$

$$\check{T}_{F,L} = T_{F,L}\Big|_{\check{\Delta}_{F,L};D_0=\check{\gamma}_L} \le T_{F,L} \equiv \frac{(\Delta_{F,L}D)A_L P_L}{(1-\Delta_{F,L}D)} \le T_{F,L}\Big|_{\hat{\Delta}_{F,L};D_0=\hat{\gamma}_L} = \hat{T}_{F,L} \tag{8}$$

$$\check{T}_{B,L} = T_{B,L}\Big|_{\check{\Delta}_{B,L};D_0=\check{\gamma}_L} \le T_{B,L} \equiv \frac{\partial}{\partial N}\frac{(\Delta_{B,L}DN(\kappa+1)-\kappa)B_L P_L N^{\kappa+1}}{(\kappa+1)(1-\Delta_{B,L}DN)}\Big|_{N=1} \le T_{B,L}\Big|_{\hat{\Delta}_{B,L};D_0=\hat{\gamma}_L} = \hat{T}_{B,L} \tag{9}$$

Weights and information bits are enumerated by search, and their ratios are used to determine acceptable values for $\check{\Delta}_L$ and $\hat{\Delta}_L$. Table 2 shows the Union bounds on FEP and BEP for a (2,1,10) and a (3,1,10) code. The region of convergence for these codes, determined from the weight enumeration, is p less than 0.044 and 0.095.

p	L	UB FEP	% err. in UB	L	UB BEP	% err. in UB	p	L	UB FEP	% err. UB	L	UB BEP	% err. in UB
0.01	16	0.401^{-9}	0.5	18	0.193^{-8}	0.3	0.02	24	0.950^{-12}	0.8	26	0.376^{-11}	0.3
0.02	18	0.614^{-7}	0.9	22	0.338^{-6}	0.6	0.04	28	0.193^{-8}	0.7	30	0.828^{-8}	0.5
0.03	24	0.144^{-5}	0.8	30	0.104^{-4}	1.0	0.06	34	0.184^{-6}	0.4	36	0.939^{-6}	0.9
0.04	54	0.234^{-4}	0.9	78	0.357^{-3}	0.9	0.08	48	0.659^{-5}	1.0	66	0.522^{-4}	0.9

(a) (2,1,10) code, g = (4672,7542)   (b) (3,1,10) code, g = (4726,5562,6372)

Table 2: Union bounds for FEP and BEP using first order bounds for the tail.

IV. DISCUSSION

This paper proposes a new approach for estimating the performance of convolutional codes with arbitrary tightness. Examples for the Union bound on FEP and BEP for two codes have been presented. Bifurcation enables estimation to a desired accuracy. However, the tail still remains the more important and difficult part. Simplifying its estimate makes the approach computationally feasible for several longer codes.

Since 1979, attention has focused on estimation of FEP and BEP for convolutional codes. The approach proposed here is quite general, and can also be applied to estimate FEP and BEP directly. This would require different approaches to bifurcation, and different bounding treatment for the tails. FEP is the topic of the next publication [7]. For discussion of the results in section III, the reader is referred to [6,7].

V. REFERENCES

1. A.J. VITERBI, 'Convolutional codes and their performance in communication systems', IEEE Trans. Commun. Technol., vol. COM-19, pp. 751 - 772, Oct. 1971.
2. L. VAN DE MEEBERG, 'A tightened upper bound on the error probability of binary convolutional codes with Viterbi decoding', IEEE Trans. Inform. Theory, vol IT-20, pp. 389 - 391, May 1974.
3. M. CEDERVALL, R. JOHANNESON, K.Sh. ZIGANGIROV, 'A new upper bound on the first event error probability for maximum-likelihood decoding of fixed binary convolutional codes', IEEE Trans. Inform. Theory, vol IT-30, Sep. 1984.
4. K.A. POST, 'Explicit evaluation of Viterbi's union bounds on convolutional code performance for the binary symmetric channel', IEEE Trans. Inform. Theory, vol IT- 23, pp. 403 - 404, May 1977.
5. S. LIN, D.J. COSTELLO, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, New Jersey, 1983.
6. A.S. ARORA, 'Bifurcation to estimate the performance of convolutional codes with arbitrary tightness', Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, IN., 1988.
7. M.V. BURNASHEV, A.S. ARORA, D.L. COHN, 'Arbitrarily tight estimate for the first event error probability of convolutional codes through bifurcation.', to be submitted to IEEE Trans. Inform. Theory.

CONSTRUCTION OF GOOD DISTANCE PROFILE TRELLIS CODES FOR SEQUENTIAL DECODING

SANKAR S.MALLADI
Dept. of Elect. Engr.
Univ. of Notre Dame
Notre Dame, IN 46556

DANIEL J.COSTELLO Jr.
Dept. of Elect. Engr.
Univ. of Notre Dame
Notre Dame, IN 46556

HENDRIK C.FERREIRA
Laboratory for Cybernetics
Rand Afrikaans Univ.
P.O.Box 524, Johannesburg
2000 South Africa

ABSTRACT

In this paper, we construct trellis codes with rate k/k+1 systematic binary convolutional codes and multi level / phase signal sets which have good distance profiles using a nested step by step search algorithm. We then simulate the Fano algotithm to study the effects of the distance profile on the computational load of a sequential decoder. The results of two such decoding simulations are presented.

INTRODUCTION

In some practical applications where the available signal energy is relatively high, a Viterbi decoder at the receiving end of the communication system would be very inefficient. This is because, even for high Signal to Noise Ratio(SNR), it still performs 2^M computations per decoded information block, where M is the encoder memory. On the other hand, a sequential decoder, whose computational speed is adaptable to varying noise levels, would decode much faster on a high SNR channel.

Another advantage of using a sequential decoder is that the decoding speed is independent of the encoder memory, whereas the number of computations performed by a Viterbi decoder increases exponentially as the encoder memory increases. This allows long constraint length codes to be used with a sequential decoder.

The performance of a sequential decoder depends on the Column Distance Function(CDF) of the code being used. Also, it is the initial part of the CDF, the Distance Profile, which affects the performance more than the latter part. These results for binary convolutional codes were first discovered by Chevillat and Costello[1].

DEFINITIONS

A rate k/k+1 systematic binary convolutional encoder of the form shown in figure[1(a)] can be described by a binary generator vector G(i), where

$$G(i) = [\ g^1(i), \ldots\ldots, g^k(i)\ ]^T, \qquad i = 0,1,\ldots\ldots,M,$$

and $g^h(i)$, $1 \leq h \leq k$, is equal to 0 if there is no connection to a mod-2 adder and equal to 1 if there is a connection and T stands for the transpose of the matrix. Figure[1(a)] shows the straight forward form of a rate2/3 systematic binary convolutional encoder and fig[1(b)] shows an alternate form the same encoder. Here, $G(0) = [0\ 0]^T$, $G(1) = [1\ 0]^T$, $G(2) = [0\ 0]^T$, $G(3) = [1\ 0]^T$, and $G(4) = [1\ 1]^T$. The encoder in figure[1(b)] can now be described by:

This work was supported by NASA grant NAG5-557.

$$G = \begin{bmatrix} g^{(1)} \\ \cdot \\ \cdot \\ g^{(k)} \end{bmatrix} = [\ G(M)\ .....G(0)\ ]$$

The CDF of a trellis code can be defined in the same way as for a binary convolutional code by replacing the Hamming weights by the Euclidean weights[2]. The CDF of order i, d_i , is defined as

$$d_i = \operatorname*{Min}_{\underline{x}_0 \neq \underline{x}'_0} \left[\sum_{n=0}^{i} \mid a(\underline{z}_n) - a(\underline{z}'_n) \mid^2 \right]$$

$$\leq \operatorname*{Min}_{\{\underline{e}_n\} \neq \{\underline{0}\}} \left[\sum_{n=0}^{i} \Delta^2_{q(\underline{e}_n)} \right]$$

where $\{\underline{x}_n\}$ and $\{\underline{x}'_n\}$ are two distinct information sequences, $\{\underline{z}_n\}$ and $\{\underline{z}'_n\} = \{\underline{z}_n + \underline{e}_n\}$ are the corresponding code sequences , $\{\underline{e}_n\}$ is the error sequence, $\{a(\underline{z}_n)\}$ and $\{a(\underline{z}'_n)\}$ are the corresponding signal sequences, $\Delta^2_{q(\underline{e}_n)}$ are the Euclidean weights, and $q(\underline{e}_n)$ is the number of trailing zeros in $\{\underline{e}_n\}$. The distance profile of a trellis code can be defined as its CDF over only the first constraint length, i.e., $(d_0, d_1, d_2,..,d_M)$.

A trellis code is said to have a distance profile $(d_0,...d_M)$ superior to the distance profile $(d'_0,...,d'_M)$ of another code of the same memory order M if for some p, $0 \leq p \leq M$,

$$\begin{aligned} d_i &= d'_i, \quad i = 0,1,2,....,p-1 \\ &> d'_i, \quad i = p. \end{aligned}$$

Ungerboeck[3] constructed trellis codes for a Viterbi decoder. Later, Porath & Aulin [4] constructed mostly optimum trellis codes with either the same or a better d_{free}. In both cases the goal was to optimize the free distance, although in the latter case the algorithm also tries to acheive a good CDF. In our search for codes, the only goal was to optimize the distance profile. A combining algorithm to construct codes with good distance profile is also described in [4].

THE CODE SEARCH ALGORITHM

We employed a nested step by step search algorithm to construct codes with a good distance profile. Lin and Lyne[5] were the first to develop this approach which was also later used by Costello[6] and Hagenauer[7]. The code construction algorithm is outlined below.

Step 1:

Choose the initial k-bit connection vector which results in the maximum value of d_0. (It turns out that this vector is always G(0) = {0} for trellis codes.)

Step 2:

Store the information bits for the set of 2^k (one branch) paths along with their distances and the initial k-bit connection vector along with the distance d_0 and set i = 0, where i is the length of the shift registers.

Step 3:

Increment i and extend all the branches stored using each of the 2^k possible combinations of connection vectors G(i) and compute their distances d_i.

Step 4:

Choose the G(i) that results in the maximium growth of the distance profile. In case of a tie, the vector which resulted in the minimum number of paths with distance d_i is chosen.

Step 5:

Store the information bits for all paths of length i+1 branches along with their distances and the connection vector G(i) along with the distance d_i. If i = M, stop. Otherwise, repeat Step 3 until i = M, the encoder memory.

It should be noted that the number of paths whose distance must be computed increases exponentially with i. For this reason, a distance cutoff value is introduced and only those paths that do not exceed this cutoff value are stored for further extension. For short constraint length codes, a reasonably high cutoff value can be used to construct the codes along with their distance profiles. The cutoff value can be estimated from the distance profiles of the shorter constraint length codes using the following lemma.

Lemma:

For rate k/k+1 trellis coded modulation schemes,

$$d_{i+1} - d_i \leq \Delta_0, \quad i = 1,2,...,M-1.$$

Proof:

Consider the minimal form of the encoder shown in figure [1(b)]. Let d_j be the minimum distance at the j th stage.

Case 1. The content of the M th stage of the shift register is a 1. Then the signal selected would be from the subset $\{ S_1, S_3, ..., S_{2(k+1)-1} \}$, which would result in an increase of Δ_0 at the (j+1) th stage.

Case 2. The content of the M th stage of the shift register is a 0. Then, the signal selected would be from the subset $\{ S_0, S_2, ..., S_{2(k+1)-2} \}$, which would result in no increase at the (j+1) th stage.

The algorithm employs a straight forward binary encoder to construct the codes. The generators given in the tables[1&2] are for the form of the encoder shown in figure[1(b)].

The mapping of the k+1 bit binary code symbols z_n into the signal points $a(z_n)$ was done by using the set-partitioning method described in [3]. Trellis codes using rate 2/3(8PSK) and 3/4(16QAM) systematic binary convolutional codes were constructed. Codes using four and eight dimensional modulation schemes were also constructed.

We show in Tables[1&2] a list of trellis codes with rate 2/3 and rate 3/4 systematic binary convolutional codes using 8-PSK and 16 QAM signal constellations, respectively. Figure[2] compares the distance profiles of two rate 2/3,constraint length 9 codes - one was constructed with the step by step algorithm presented here to optimize the distance profile and the other was constructed by Ungerboeck[3] to optimize the free distance. Figure[3] compares the distance profiles of two other codes, a good distance profile code constructed here and a mostly optimal code constructed with the combining algorithm[8], both of which are rate 2/3, constraint length 13 codes.

DECODING SIMULATIONS

To study how the distance profile affects the decoding speed or the number of computations required per frame by a sequential decoder, we simulated a Fano decoder and decoded noisy frames generated by using the rate 2/3 constraint length 13 code and the rate 2/3, constraint length 9 code constructed with the step by step algorithm. We then used the mostly optimal(MO) code of rate 2/3, constraint length 13 and the rate 2/3 constraint length 9 code constructed by Ungerboeck(UG).

Each simulation involved generating a thousand random sequences of 1's and 0's, encoding, and mapping the code words into an 8-PSK signal set. Each sequence was a frame of 128 symbols. A Gaussian channel was simulated to transmit these signal points. Hard decisions were made by the Fano decoder based on the optimum decision regions shown in figure[4]. Each forward movement of the decoder was counted as one computation.The metric used was not an optimum metric, but was good enough to compare the computational distributions of the different codes. The bit metric used for decoding binary convolutional codes was used as the symbol metric for trellis codes:

$$M[r(\underline{z}_n)|a(\underline{z}_n)] = \log_2 [\, P(r(\underline{z}_n)|a(\underline{z}_n)) \,/\, P(r(\underline{z}_n))] - R,$$

where, $r(\underline{z}_n)$ is a received symbol, $a(\underline{z}_n)$ is a transmitted symbol, and R is the code rate.

The results of these simulations at SNRs of 11dB and 10dB, respectively,are shown in figures [5&6]. C is the number of computations required per frame. The figures show the probability that C is greater than or equal to a number N plotted against N. The results obtained generally agree with the results previously found using binary convolutional codes[1]. The code with the better distance profile performed fewer computations on the average.

CONCLUSIONS

The results clearly indicate that for sequential decoding it is important to have trellis codes with a rapidly increasing distance profile. Also, since the computational performance of the decoder does not depend on the constraint length of the code, long constraint length codes which guarantee a large d_{free} can be used. This makes the free distance a secondary criterion when designing trellis codes for sequential decoding.

References:

1. P.R.Chevillat and D.J.Costello,Jr., "Distance and Computation in Sequential Decoding", IEEE Trans. Commun., COM-24, pp.440-447, April 1976.

2. G.Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets. Part II: State of the Art", IEEE Commun Mag., vol.25, No.2,pp.12-21, Feb. 1987.

3. G.Ungerboeck, "Channel coding with Multi-level/Phase Signals", IEEE Trans. Inform. Theory, vol.IT-28, pp.55-67, Jan. 1982.

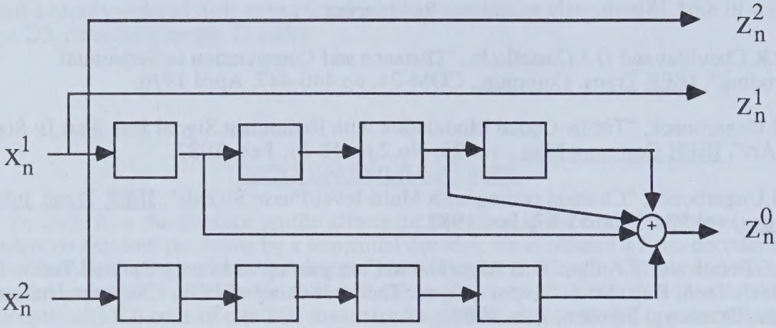
4. J.E.Porath and T.Aulin, "Fast Algorithmic Construction of Mostly Optimal Trellis Codes", Tech. Rep. No.5, Div. of Inform. Theory, School of ECE, Chalmers Univ. of Tech., Goteborg, Sweden, Feb. 1987.

5. S.Lin and H.Lyne, "Some results on Binary Convolutional Code Generators", IEEE Trans. Inform. Theory, vol.IT-13, pp.134-139, Jan. 1967.

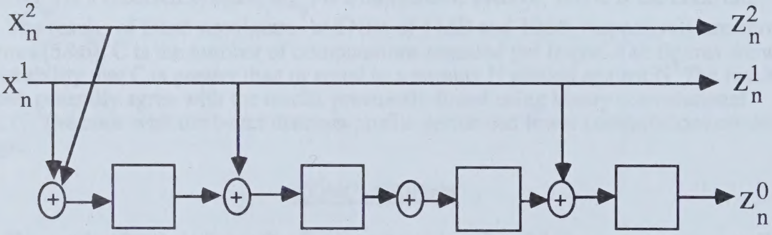
6. D.J.Costello,Jr., "A Construction Technique for Random Error Correcting Convolutional Codes", IEEE Trans. Inform. Theory, vol.IT-15, pp.631-636, Sept. 1969.

7. J.Hagenauer, "High Rate Convolutional Codes with Good Distance Profiles", IEEE Trans. Inform. Theory, vol.IT-23, pp.615-618, Sept. 1977.

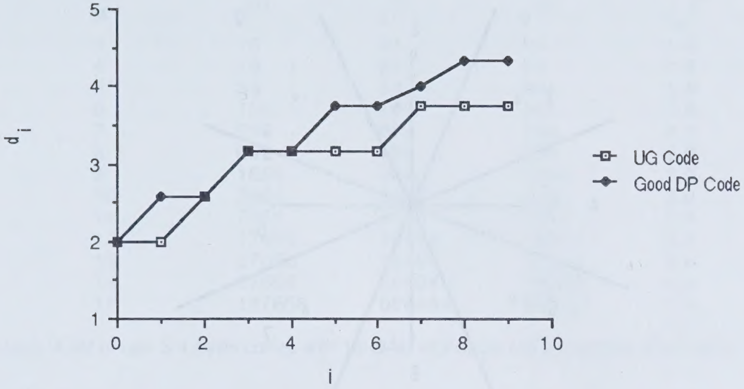
8. G.J.Pottie and D.P.Taylor, "A Comparision of Reduced Complexity Decoding Algorithms for Trellis Codes", CRL Rep. No.190, Faculty of Engr., McMaster Univ., Hamilton, Ontario, Canada, March 1988.


Z_n^2
Z_n^1
X_n^1
+
Z_n^0
X_n^2

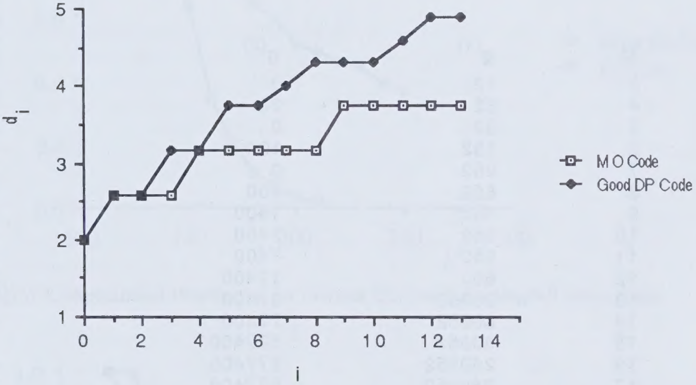
Fig[1(a)]. A straight forward realization of a rate 2/3 constraint length 4 convolutional code.


X_n^2
Z_n^2
X_n^1
Z_n^1
+
+
+
+
Z_n^0

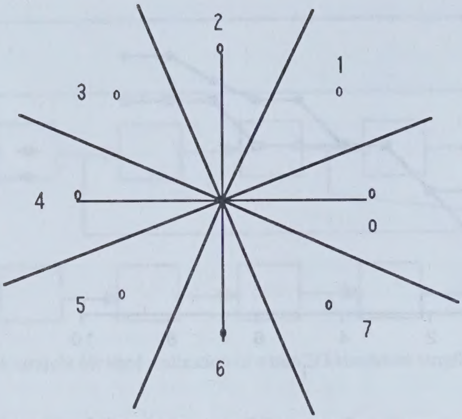
Fig[1(b)]. An alternate form of the encoder in figure[1(a)].



Fig[2]. Distance profiles of two rate 2/3, constraint length 9 trellis codes.



Fig[3]. Distance profiles of two rate 2/3, constraint length 13 trellis codes.



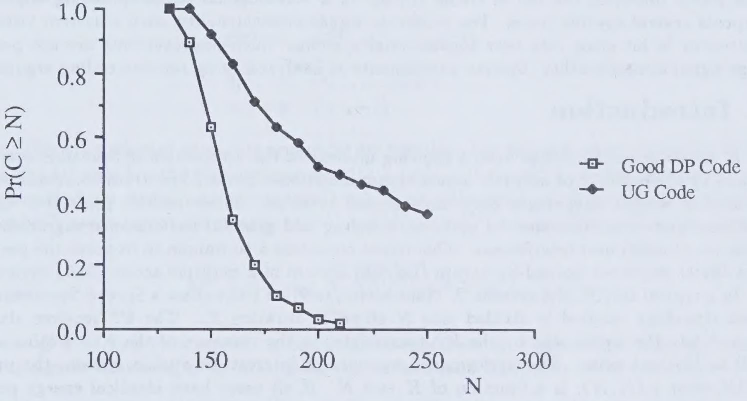
Fig[4]. 8-PSK signal set with the decision regions.

M	$g^{(1)}$	$g^{(2)}$	d_M^2
3	12	0	3.17
4	32	20	3.17
5	52	0	3.74
6	152	100	3.74
7	252	0	4.00
8	652	400	4.32
9	652	1400	4.58
10	652	3400	4.90
11	652	7400	4.90
12	652	17400	5.16
13	20652	37400	5.48
14	60652	77400	5.48
15	60652	177400	5.48
16	260652	177400	5.48
17	660652	577400	5.48
18	1660652	577400	5.48
19	3660652	577400	5.74
20	7660652	577400	5.74
21	3660652	10577400	6.02
22	23660652	10577400	6.02
23	63660652	10577400	6.02
24	163660652	110577400	6.33
25	163660652	210577400	6.33

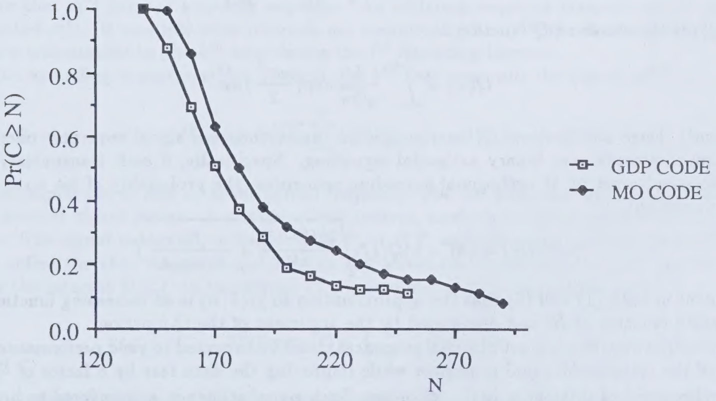
Table[1]. A list of rate 2/3 trellis codes with 8-PSK signal set.(All generators are in octal.)

M	$g^{(1)}$	$g^{(2)}$	$g^{(3)}$	$d_M{}^2$
3	16	04	04	1.2
4	16	24	24	1.6
5	56	64	64	1.6
6	156	164	164	1.6
7	256	064	264	2.0
8	656	464	364	2.0
9	1656	0464	1364	2.0
10	3656	2464	1364	2.0
11	7656	6464	5364	2.4
12	17656	16464	15364	2.4
13	37656	16464	15364	2.4
14	37656	56464	15364	2.4
15	137656	056464	015364	2.4

Table[2]. A list of rate 3/4 trellis codes with 16-QAM signal set.(All generators are in octal.)



Fig[5]. Computational Distribution for two rate 2/3, constraint length 9 trellis codes.



Fig[6]. Computational Distribution for two rate 2/3, constraint length 13 trellis codes.

The Application of Trellis Coding Techniques to Direct-Sequence Spread-Spectrum Multiple Access Communications.

WAYNE E. STARK
BRIAN D. WOERNER
The Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109

Abstract

This paper considers a Direct-Sequence Spread-Spectrum Multiple Access communication system. Previous work has considered the use of antipodal and orthogonal sets of spreading sequences. This paper considers the use of trellis coding on a biorthogonal set of spreading sequences and proposes several specific codes. For moderate signal dimensionality, such a system yields an improvement in bit error rate over conventional systems. Such improvements are not possible for large signal dimensionality. System performance is analyzed using random coding arguments.

1. Introduction

In recent years, there has been a growing interest in the application of Spread-Spectrum techniques to the problem of multiple access communications. Spread-Spectrum communications systems offer several advantages over conventional systems. These include protection against intentional jamming, resistance to multi-path fading and graceful performance degradation in the presence of multi-user interference. This paper considers a technique to improve the performance of a Direct-Sequence Spread-Spectrum (DS/SS) system in a multiple access (MA) environment.

In a typical DS/SS MA system, K transmitter/receiver pairs share a Spread-Spectrum channel. Each signalling interval is divided into N chips of duration T_c. The k^{th} receiver attempts to demodulate the signal sent by the k^{th} transmitter in the presence of the $K-1$ other signals as well as thermal noise. The performance measure of interest for such a system, the probability of bit error $p_b(K;N)$, is a function of K and N. If all users have identical energy per bit E_b, it is well known [1] that the probability of bit error for a binary antipodal DS/SS MA system is approximately

$$p_b(K;N) = Q(1/\sqrt{\frac{K-1}{3N}+\frac{N_o}{2E_b}}), \tag{1}$$

where $Q()$ is the standard Q-function.

$$Q(\alpha) = \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(\frac{-x^2}{2})\, dx \tag{2}$$

Recently, Enge and Sarwate [2] have suggested that orthogonal signal sequences offer performance improvements over binary antipodal signalling. Specifically, if each transmitter/receiver pair is assigned a set of M orthogonal spreading sequences, the probability of bit error may be approximated by

$$p_b(K;N) = (M-1)Q(1/\sqrt{\frac{2(K-1)}{3N}+\frac{N_o}{E_b log_2 M}}). \tag{3}$$

It is evident in both (1) and (3) that the approximation to $p_b(k;n)$ is an increasing function of K, a decreasing function of N, and dominated by the argument of the Q-function.

Obviously, a biorthogonal set of signal sequences could be expected to yield performance similar to that of the orthogonal signal sequences while improving the data rate by a factor of $\frac{\log_2 M+1}{\log_2 M}$. The novel concept of this paper is the following. Each signal sequence is considered to be a point in a signal constellation, and Ungerboeck's method of set partioning [3] is applied to the case of biorthogonal signal sequences. It is shown in the sequel that the resulting trellis coded DS/SS

MA system performs substantially better than either binary antipodal or orthogonal signalling for moderate values of M, without increasing the bandwidth requirements.

The remainder of this paper is organized as follows. Section 2 introduces the system models. In Section 3, the new technique is described, several simple trelllis codes are presented and it is shown that good codes do not exist for large values of M. Section 4 analyzes the performance of these trellis codes in a multiple access enviroment, considering the case of random coding. Section 5 concludes this paper.

2. System Models

In this section, the system models and terminology are described. A prototypical DS/SS MA system is illustrated in Figure 1. There are K transmitter/receiver pairs sharing a common channel. During a signalling interval of duration T, each transmitter receives m new data bits and selects one of Q possible signals to send.

In DS/SS, the bandwidth spreading is uniquely determined by dividing each signal into N 'chips' of length T_c, where $T = NT_c$. The transmitted signal is uniquely determined by the sequence of chips or 'spreading sequence' used. Each of the K users posesses a unique set of Q spreading sequences. The k^{th} such set is denoted $S^{(k)}$. The sequences in this set may be thought of as vectors whose members take on values in $\{+1,-1\}^N$.

$$\begin{aligned} S^{(k)} &= \{\mathbf{s}_1^{(k)}, \mathbf{s}_2^{(k)}, \ldots, \mathbf{s}_Q^{(k)}\}, \\ \mathbf{s}_i^{(k)} &= (s_{i,1}^{(k)}, \ldots, s_{i,N}^{(k)}), \\ s_{i,j} &\in \{+1,-1\}. \end{aligned} \tag{4}$$

Many different choices of spreading sequences are possible. For example when antipodal DS/SS is chosen, the k^{th} user has the binary signal set $S^{(k)} = \{\mathbf{s}_1^{(k)}, \mathbf{s}_2^{(k)} = -\mathbf{s}_1^{(k)}\}$. In this case $m = 1$ data bit is transmitted per signalling interval and $Q = 2$. If an orthogonal set of spreading sequences is chosen, the k^{th} user's signal set consists of $S^{(k)} = \{\mathbf{s}_1^{(k)}, \ldots, \mathbf{s}_M^{(k)}\}$ where $\mathbf{s}_i^{(k)} \perp \mathbf{s}_j^{(k)}$ in the sense of

$$\mathbf{s}_i^{(k)} \perp \mathbf{s}_j^{(k)} \Longleftrightarrow \sum_{l=1}^{N} s_{i,l}^{(k)} s_{j,l}^{(k)} = 0, \forall i \neq j. \tag{5}$$

In this case $Q = M$ and $m = \log_2 M$. A biorthogonal signal set is considered in this paper. Each user has Q signals consisting of M orthogonal signals and their antipodes as described by

$$S^{(k)} = \{\mathbf{s}_1^{(k)}, \ldots, \mathbf{s}_Q^{(k)}\} = \{\mathbf{s}_1^{(k)}, \ldots, \mathbf{s}_M^{(k)}, -\mathbf{s}_1^{(k)}, \ldots, -\mathbf{s}_M^{(k)}\} \tag{6}$$

where $\mathbf{s}_i^{(k)} \perp \mathbf{s}_j^{(k)}$ whenever $i \neq j, j \pm M$ and $\mathbf{s}_i^{(k)} = -\mathbf{s}_i^{(k)}$ if $i = j \pm M$.

Note that $\mathbf{s}_1^{(k)}$ denotes a specific sequence. An arbitrary sequence transmitted by user k will be denoted $\mathbf{s}^{(k)}$. If multiple time intervals are considered, then $\mathbf{s}^{(k)}[l]$ will denote the spreading sequence transmitted by the k^{th} user during the l^{th} signalling interval.

If the spreading sequence $\mathbf{s}^{(k)}$ is selected, the k^{th} user transmits the signal $s(\mathbf{s}^{(k)}, t)$.

$$s(\mathbf{s}^{(k)}, t) = \sqrt{2P_s} \sum_{n=1}^{N} s_n^{(k)} \Gamma[t - (n-1)T_c] \cos(\omega_c t) \tag{7}$$

P_s is the signal power and ω_c is the carrier frequency. For our purposes, all users are assumed to have identical signal power. A similar, albeit tedious, analysis could be carried out for unequal powers. The signal energy E_s is found to be $E_s = P_s T$, and the energy per bit E_b equals E_s/m. $\Gamma(\cdot)$ is called the chip waveform and may be any waveform which has energy T_c and is non-zero only in the interval $[0,T_c)$. In the sequal, $\Gamma(\cdot)$ is assumed to be a rectangular pulse.

$$\Gamma(t) = p_T(t) \equiv \begin{cases} 1 & t \in [0, T_c) \\ 0 & t \notin [0, T_c) \end{cases} \tag{8}$$

The transmitted signal $s(\mathbf{s}^{(k)}, t)$ passes through a channel. In the channel, the k^{th} signal is delayed by τ_k, where τ_k is a random variable uniformly distributed on $[0,T)$. Each carrier signal also has a random phase associated with it, but this does not effect the analysis and is omitted for the sake of notational simplicity. The channel output $r(t)$ is the sum of the K delayed signals and thermal noise $n(t)$. The thermal noise is modeled as AWGN with (two-sided) spectral density $N_o/2$. Thus

$$r(t) = n(t) + \sum_{k=1}^{K} s(\mathbf{s}^{(k)}, t - \tau_k). \tag{9}$$

The same signal $r(t)$ is presented to each of the K receivers, assumed to be correlation receivers. A set of Q decision statistics $\{Z_q : q = 1 \ldots Q\}$ is obtained at the k^{th} receiver by correlating the received signal with the Q possible transmitted signals. That is,

$$Z_q^{(k)} = \int_{\tau_k}^{T+\tau_k} s(\mathbf{s}_q^{(k)}, t - \tau_k) r(t)\, dt. \tag{10}$$

The signal associated with the largest correlation is selected. The k^{th} receiver is able to synchronize to the k^{th} user.

3. Trellis Coding for Biorthogonal Signal Sets

In this section, a form of trellis coding on a biorthogonal set of spreading sequences is proposed. It is suggested that the use of such codes will result in performance improvements over both binary antipodal and orthogonal spreading sequences. An analysis of error probability is offered in Section 4. It is also shown that good codes of this class will not exist for large dimensionality M.

An N chip spreading sequence defines a point in $\Re^N$. It follows that any set of M orthogonal spreading sequences is contained in an M dimensional subspace of $\Re^N$. This generates via (7) a set of orthogonal signals. If the antipodes of each spreading sequence are also included in the signal set, this new signal set consists of $2M$ signals in M dimensions.

In the absence of coding, this biorthogonal signal set is a trivial improvement over the scheme proposed in [2]. Doubling the number of signals allows the transmission of one extra bit per signalling interval. Thus the signalling interval and signal energy can be increased by a factor of $\frac{1+\log_2 M}{\log_2 M}$ with identical data rate. The performance of this scheme is approximated by

$$p_b(K;N) = (2M-2)Q\left(1/\sqrt{\frac{2(K-1)}{3N} + \frac{N_o}{E_b(1+\log_2 M)}}\right). \tag{11}$$

The performance improvement obtainable from this scheme is limited, because each increase in signal energy also represents an increase in the multiple access interference. There is no fundamental improvement in the separation between signals.

However, in this paper it is proposed that the extra bit obtained by doubling the signal set be used to provide redundancy through a form of trellis coding. Following the method of set partitioning, an M dimensional orthogonal signal constellation is expanded into a biorthogonal signal constellation containing $2M$ signals. Each point in this signal constellation defines a specific spreading sequence. Each point has $2M-2$ neighbors at distance $\sqrt{2E_s}$ and one neighbor at distance $2\sqrt{E_s}$. Using Ungerboeck's set partitioning techniques to guide the search [3], trellis codes which exploit this distance structure may be found.

A trellis code may be implemented via a finite state encoder and a soft decision Viterbi decoder. This paper will represent a trellis code by its underlying trellis. The trellis is an array of nodes, partially connected in a repeating pattern of transitions. The passage of successive time intervals is indicated by transitions to the right. The current state of the encoder is indicated by the vertical position of the current node. Each node has 2^m transitions emanating to the right; one of these is selected by the m bits of input data from the current time interval. The signal which labels the selected transition is sent and the encoder's new state is given by the node at the right end of the transition. In the figures of this paper, the labels associated with transitions from a given state are listed to the left of that state in vertically descending order.

The advantage of trellis coding over uncoded modulation is that the minimum Euclidean distance between distinct paths through the trellis, d_{free}, may be made larger than the minimum Euclidean distance separating distinct members of the signal set, d_{min}. Note that this d_{min} is a parameter of the original uncoded signal set and not the expanded signal constelation on which the trellis code is based. The ratio of the squares of the distances, known as the coding gain G, is measured in decibels, and represents the energy savings for identical error performance at high signal to noise ratio over an AWGN channel.

$$G = 10\log_{10}(\frac{d_{free}}{d_{min}})^2 \tag{12}$$

The maximum number of distinct paths at minimum distance d_{free} from any path which separate from that path at a single node is known as N_{free}.

Now consider the example of Figure 2. A 2 dimensional orthogonal signal set is expanded into a biorthogonal signal set of $Q = 4$ signals. Set partitioning results in 2 and 4 state trellis codes with d_{free} of $\sqrt{6E_s}$ and $\sqrt{8E_s}$, respectively, and coding gaine of 4.8 dB and 6.0 dB, respectively. N_{free} equals one for both codes. This coding gain is achieved in the same signal dimensionality $M = 2$ and with the same data rate $m = 1$ bit/interval as the orthogonal signal set. Figures 3 and 4 illustrate trellis codes for the cases of $m = 2$ and $m = 3$ respectively.

Note that larger data rates result in diminishing coding gains and higher complexity to achieve them. In fact the coding gain must drop to zero for large signal dimensionality.

$$\lim_{M\to\infty} \max_{\{trellis\ codes\}} G = 0 \tag{13}$$

This proof of this is simple and follows from the identical assymptotic energy efficiency of orthogonal and biorthogonal signal sets. Assymptotically as M approaches infinity, the orthogonal signal set achieves capacity for the AWGN channel. As a result, we cannot expect to improve upon the performance of orthogonal signals for large M. Therefore, gain of the trellis code must approach zero.

4. Performance Analysis

In this section, the performance analysis of trellis codes for DS/SS MA channels is considered. The performance measure of interest is $p_b(K;N)$. A reasonably tight upper bound is the probability of an error event $p_e(K;N)$ along any arbitrary path p through the trellis. An error event is any path $\tilde{p}$ which diverges from p for a finite number of intervals beginning with the first one. The probability of an error event occuring equals the probability that the decision statistic $Z_{\tilde{p}}$ from a correlation receiver is greater than Z_p. The union bound is applied to the sum of individual error events [6].

$$p_b(K;N) \leq p_e(K;N) = \sum_{\{\tilde{p}:\tilde{p}\neq p\}} Pr\{Z_{\tilde{p}} > Z_p\} \tag{14}$$

The summation in (14) is over a potentially infinite number of paths. However, for high SNR, only the error events at the minimum distance d_{free} are significant.

$$p_b(K;N) \leq \sum_{\{\tilde{p}:\|p-\tilde{p}\|=d_{free}\}} Pr\{Z_{\tilde{p}} > Z_p\} \tag{15}$$

Therefore it is of interest to calculate the probability of a single error event. Examine Figure 5. Such an error event begins during the first interval of interest and has duration L_e intervals. The vectors $\mathbf{s}_p$ and $\mathbf{s}_{\tilde{p}}$ are concatenations of the spreading sequence vectors during the L_e intervals for which the paths p and $\tilde{p}$ diverge.

$$\mathbf{s}_p = (\mathbf{s}^{(1)}[1], \mathbf{s}^{(1)}[2], \ldots, \mathbf{s}^{(1)}[L_e])$$
$$\tilde{\mathbf{s}}_p = (\tilde{\mathbf{s}}^{(1)}[1], \tilde{\mathbf{s}}^{(1)}[2], \ldots, \tilde{\mathbf{s}}^{(1)}[L_e]) \tag{16}$$

The signal associated with each path in the interval $[0, L_eT)$ is thus desribed by

$$s(\mathbf{s}_p, t) = \sqrt{2P_s}\sum_{n=1}^{L_eN} s_{p,n}\Gamma[t-(n-1)T_c]\cos(\omega_c t),$$
$$s(\mathbf{s}_{\tilde{p}}, t) = \sqrt{2P_s}\sum_{n=1}^{L_eN} s_{\tilde{p},n}\Gamma[t-(n-1)T_c]\cos(\omega_c t). \tag{17}$$

Now let $S_k = \tau_k - \gamma_k T_c$ and define the partial waveform crosscorrelations $R_1(S_k)$ and $R_2(S_k)$.

$$\begin{aligned} R_1(S_k) &= \int_0^{S_k} \Gamma(t - S_k + T_c)\Gamma(t)\,dt \\ R_2(S_k) &= \int_{S_k}^{T_c} \Gamma(t - S_k)\Gamma(t)\,dt \end{aligned} \tag{28}$$

For the special case of $\Gamma(\cdot)$ a rectangular pulse, $R_1(S_k) = S_k$ and $R_2(S_k) = T_c - S_k$. Using definition (28), (27) can be simplified to

$$n_I^{(k)} = P_s \cos(\omega_c \tau_k)\{\sum_{m=1}^{L_e N} a_{m-1}^{(k)} s_{p,m} R_1(S_k) + \sum_{m=1}^{L_e N} a_m^{(k)} s_{p,m} R_2(S_k)\}. \tag{29}$$

A random coding argument may now be employed [5]. Let $X_m = a_{m-1}^{(k)} s_{p,m}$ and $Y_m = a_m^{(k)} s_{p,m}$. Assume that X_m and Y_m are independent, equiprobable binary random variables taking values $\{+1, -1\}$. This is equivalent to assuming a low cross-correlation between different user's spreading sequences. This results in

$$n_I^{(k)} = P_s \cos(\omega_c \tau_k)\{R_1(S_k) \sum_{m=1}^{L_e N} X_m + R_2(S_k) \sum_{m=1}^{L_e N} Y_m\}. \tag{30}$$

If N becomes large and the Central Limit Thereom [4] allows us to approximate both $\sum_{m=1}^{L_e N} X_m$ and $\sum_{m=1}^{L_e N} Y_m$ as Gaussian random variables with mean zero and variance $L_e N$. As a result, $n_I^{(k)}$ is also approximated by a Gaussian random variable with mean zero and conditional variance $\sigma^2_{n_I^{(k)}|\tau_k}$.

$$\sigma^2_{n_I^{(k)}|\tau_k} = L_e N P_s^2 \cos^2(\omega_c \tau_k)\{[R_1(S_k)]^2 + [R_2(S_k)]^2\} \tag{31}$$

At this point the variance is still conditioned on the uniformly distributed random delay τ_k. The unconditional variance may be found by means of the iterated expectation formula [4].

$$\begin{aligned} \sigma^2_{n_I^{(k)}} &= E_{\tau_k}[var(n_I^{(k)} \mid \tau_k)] + var_{\tau_k}[E(n_I^{(k)} \mid \tau_k)] \\ &= 1/T \int_0^T \sigma^2_{n_I^{(k)}|\tau_k}\, d\tau_k \\ &= \frac{2 L_e E_s^2}{3N} \end{aligned} \tag{32}$$

Computing the integral of (32) and again dropping double frequency terms, it is found that $n_I^{(k)}$ is Gaussian with mean zero and variance $2L_e E_s^2/3N$. Since the $K-1$ interferers are independent, it follows that $\sum_{k=2}^{K} \tilde{n}_I^{(k)}$ is Gaussian with identical mean and variance. By summing all of the noise components in (23), it is concluded that $\Pr\{Z_{\tilde{p}} - Z_p > 0\} = \Pr\{\eta - d^2_{free}/2 > 0\}$, where η is the Gaussian random variable described by $m_\eta = 0$ and

$$\sigma_\eta^2 = \frac{4 L_e E_s^2 (K-1)}{3N} + L_e N_o E_s. \tag{33}$$

For reasonably high SNR, only the error event at minimum distance will occur with any consistency. Therefore, the probability of bit error can be bounded by

$$p_b(K;N) = N_{free} Q(\frac{d^2_{free}}{2\sigma_\eta}). \tag{34}$$

Or equivalently,

$$p_b(K;N) = N_{free} Q(1/\sqrt{\frac{4 L_e E_s (K-1)}{3 N G_r^4} + \frac{2 L_e N_o}{m E_b G_r^4}}), \tag{35}$$

where G_r is the ratio d_{free}/d_{min}.

A correlation receiver will produce the decision statistics Z_p and $Z_{\tilde{p}}$, and the goal is to calculate the probability $Pr\{Z_{\tilde{p}} > Z_p\} = Pr\{Z_{\tilde{p}} - Z_p > 0\}$.

$$Z_p = \int_0^{L_e T} r(t)\, s(\mathbf{s}_p, t)\, dt$$

$$Z_{\tilde{p}} = \int_0^{L_e T} r(t)\, s(\mathbf{s}_{\tilde{p}}, t)\, dt \tag{18}$$

The received signal $r(t)$ consists of the transmitted signal, thermal noise and the multiple access interference from the $K-1$ other users.

$$r(t) = s(\mathbf{s}_p, t) + n_W(t) + \sum_{k=2}^{K} n_I^{(k)}(t),\ t \in [0, L_e T) \tag{19}$$

$n_I^{(k)}(\cdot)$ in (19) is the signal received from the k^{th} user during the time interval $[0, L_e T)$. Because the k^{th} user's signal is delayed with respect to the first user's by an independent, uniformly distributed random variable τ_k, $L_e + 1$ signal intervals of user k will overlap with L_e intervals of user 1. Furthermore, $NL_e + 1$ chips from user k's spreading sequence will overlap with the NL_e chips of user 1. Let γ_k be defined by

$$\gamma_k = \lfloor \tau_k / T_c \rfloor. \tag{20}$$

After the following notational simplification, the analog signal $n_I^{(k)}(t)$ may be defined in the interval $[0, L_e T)$.

$$\mathbf{a}^{(k)} = (a_0, \ldots, a_{L_e N}) = (s_{\gamma_k}^{(k)}[0], \ldots, s_N^{(k)}[0], \ldots, s_1^{(k)}[L_e], \ldots, s_{\gamma_k}^{(k)}[L_e]) \tag{21}$$

$$n_I^{(k)}(t) = \sqrt{2P_s} \sum_{n=0}^{L_e N} a_n^{(k)} \Gamma[t - \tau_k - (n - \gamma_k) T_c] \cos[\omega_c (t - \tau_k)] \tag{22}$$

It is now neccesary to compute $Z_{\tilde{p}} - Z_p$. Combining (18), (19) and (22) yields

$$\begin{aligned} Z_{\tilde{p}} - Z_p &= \int_0^{L_e T} [s(\mathbf{s}_{\tilde{p}}, t) - s(\mathbf{s}_p, t)]\, r(t)\, dt \\ &= \tilde{n}_W + \sum_{k=2}^{K} \tilde{n}_I^{(k)} - n_W - \sum_{k=2}^{K} n_I^{(k)} - (d_{free})^2/2 \end{aligned} \tag{23}$$

where

$$n_W = \int_O^{L_e T} s(\mathbf{s}_p, t)\, n_W(t)\, dt\ ,\ \tilde{n}_W = \int_0^{L_e T} s(\mathbf{s}_{\tilde{p}}, t)\, n_W(t)\, dt, \tag{24}$$

$$n_I^{(k)} = \int_0^{L_e T} s(\mathbf{s}_p, t)\, n_I^{(k)}(t)\, dt\ ,\ \tilde{n}_I^{(k)} = \int_0^{L_e T} s(\mathbf{s}_{\tilde{p}}, t)\, n_I^{(k)}(t)\, dt. \tag{25}$$

By a straightforward argument, n_W and $\tilde{n}_W$ are Gaussian random variables with mean zero and variance $L_e N_o E_s/2$.

The computation of $n_I^{(k)}$ and $\tilde{n}_I^{(k)}$ is more involved. Expand the equation of $n_I^{(k)}$ in terms of the spreading sequences.

$$n_I^{(k)} = 2P_s \int_0^{L_e T} \sum_{n=1}^{L_e N} \sum_{m=0}^{L_e N} s_{p,n} a_m^{(k)} \Gamma[t-(n-1)T_c] \Gamma[t-\tau_k-(m-\gamma_k)T_c] \cos(\omega_c t) \cos[\omega_c(t-\tau_k)]\, dt \tag{26}$$

Ignoring the double frequency components and noting the finite extent of $\Gamma(\cdot)$, (26) is simplified to

$$\begin{aligned} n_I^{(k)} &= 2P_s \cos(\omega_c \tau_k) \\ &\quad \{\sum_{n=1}^{L_e N} \int_{nT_c}^{(n-\gamma_k)T_c+\tau_k} a_{n-1}^{(k)} s_{p,n} \Gamma[t - \tau_k - (n - \gamma_k) T_c] \Gamma(t - mT_c)\, dt \\ &\quad + \sum_{m=1}^{L_e N} \int_{(m-\gamma_k)T_c+\tau_k}^{mT_c} a_m^{(k)} s_{p,m} \Gamma[t - \tau_k - (m - \gamma_k) T_c] \Gamma(t - mT_c)\, dt\}. \end{aligned} \tag{27}$$

As a typical example, consider the 4 state code used with the 8-ary biorthogonal signal set of Figure 3. This code has parameters $d_{free} = \sqrt{4E_s}$, $N_{free} = 1$, $m = 2$ and $L_e = 1$. In this case, the random coding argument leads to the following estimate of $p_b(K;N)$.

$$p_b(K;N) = Q(1/\sqrt{\frac{K-1}{12N} + \frac{N_o}{16mE_b}}) \tag{36}$$

This compares favorably with (1) and (3). This performance is also better than what could be obtained with an uncoded biorthogonal signal set.

Applying the techniques of Pursley [5], it is also possible to obtain performance results for sequences with arbitrary crosscorrelation properties. The method of Lehnert [7], may also be used to obtain better estimates of the performance. We are currently investigating the performance of the trellis codes in a multi-path fading environment.

5. Conclusions

This paper has presented an application of trellis coding techniques to the problem of DS/SS MA communications. For signal sets of moderate size, it has been shown that the use of trellis coded biorthogonal spreading sequence sets can yield performance improvements over either antipodal or orthogonal spreading sequences. This also appears to be the first work to specifically examine the use of trellis codes for biorthogonal signal sets. For large biorthogonal sets, there do not exist good trellis codes.

Acknowledgement

This work was supported by the National Science Foundation under grant ECS-8451266.

References

[1] M. B. Pursley, "Performance evaluation for phase-coded spread-spectrum multiple-access communication - part I: System analysis," *IEEE Transactions on Communications*, vol. COM-25, pp. 795-799, August 1977.

[2] P. K. Enge and D. V. Sarwate, "Spread Spectrum Multiple Access Performance of Orthogonal Codes - Linear Receivers," *IEEE Transactions on Communications*, to appear.

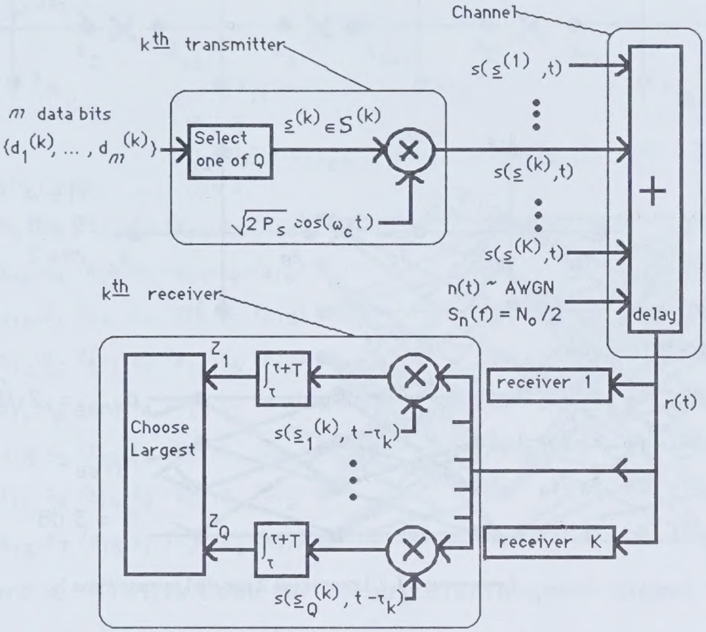
[3] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, vol. IT-30, pp. 55-67, January 1982.

[4] H. Stark and J. W. Woods, *Probability, Random Processes and Estimation for Engineers*, Englewood Cliffs: Prentice-Hall, 1986.

[5] M. B. Pursley, "Spread-Spectrum Multiple-Access Communications," *Multi-User Communication Systems*, New York: Springer-Verlag, 1981.

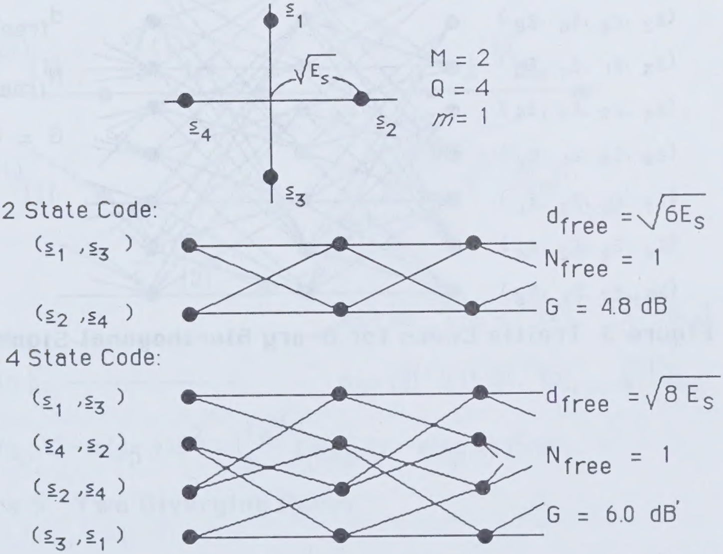
[6] E. Zehavi and J. K. Wolf, "On the Performance of Trellis Codes," *IEEE Transactions on Information Theory*, vol. IT-33, pp. 196-202, March 1987.

[7] J. S. Lehnert and M. B. Pursley, "Error probabilities for binary direct-sequence spread-spectrum communication with random signature sequences," *IEEE Transactions on Communications*, vol. COM-35, pp. 87-98, January 1987.



Channel
k th transmitter
m data bits
{d1(k), ... , dm(k)}
Select one of Q
s(k) ∈ S(k)
√2 Ps cos(ωc t)
s(s(1), t)
s(s(k), t)
s(s(K), t)
n(t) ~ AWGN
Sn(f) = No/2
+
delay
k th receiver
receiver 1
r(t)
receiver K
Choose Largest
Z1
ZQ
∫τ τ+T
s(s1(k), t − τk)
s(sQ(k), t − τk)

Figure 1. DS/SS MA Communication System.



s1
s2
s3
s4
√Es
M = 2
Q = 4
m = 1
2 State Code:
(s1, s3)
(s2, s4)
dfree = √6Es
Nfree = 1
G = 4.8 dB
4 State Code:
(s1, s3)
(s4, s2)
(s2, s4)
(s3, s1)
dfree = √8 Es
Nfree = 1
G = 6.0 dB

Figure 2. Trellis Codes for 4-ary Biorthogonal Signal Sets.

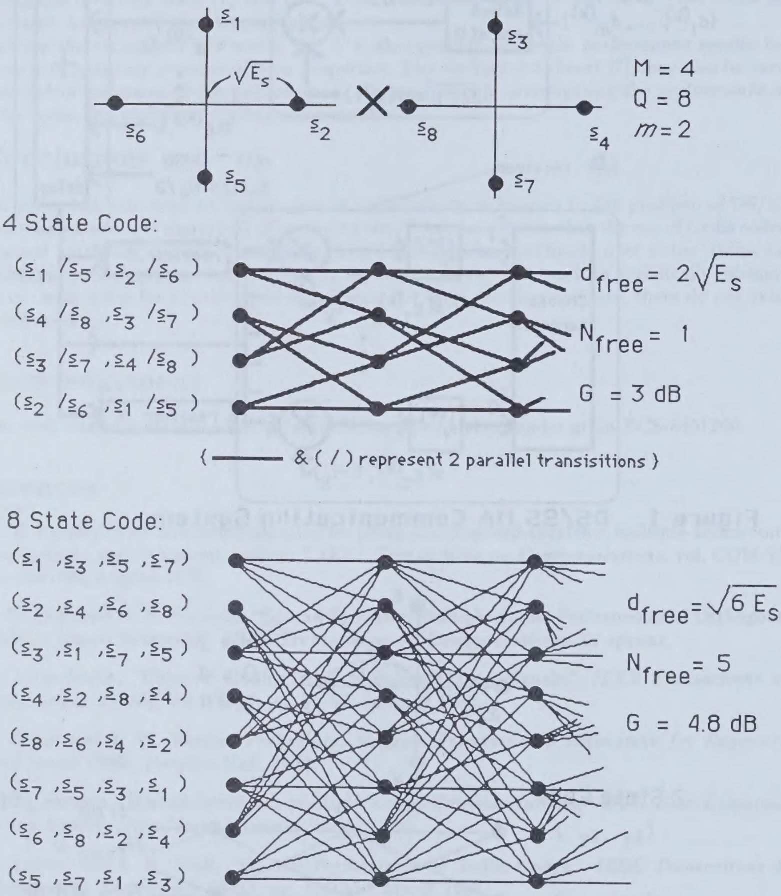

s1
s2
s5
s6
√Es
s3
s4
s7
s8
M = 4
Q = 8
m = 2
4 State Code:
(s1 /s5 ,s2 /s6)
(s4 /s8 ,s3 /s7)
(s3 /s7 ,s4 /s8)
(s2 /s6 ,s1 /s5)
dfree = 2√Es
Nfree = 1
G = 3 dB
(——— & (/) represent 2 parallel transisitions)
8 State Code:
(s1 ,s3 ,s5 ,s7)
(s2 ,s4 ,s6 ,s8)
(s3 ,s1 ,s7 ,s5)
(s4 ,s2 ,s8 ,s4)
(s8 ,s6 ,s4 ,s2)
(s7 ,s5 ,s3 ,s1)
(s6 ,s8 ,s2 ,s4)
(s5 ,s7 ,s1 ,s3)
dfree = √6 Es
Nfree = 5
G = 4.8 dB

Figure 3. Trellis Codes for 8-ary Biorthogonal Signal Sets.

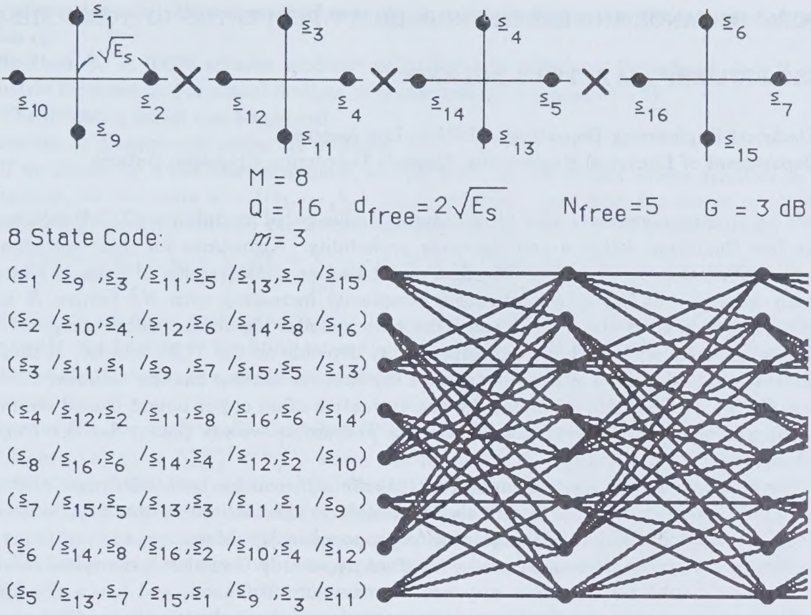

$\underline{s}_1$
$\sqrt{E_s}$
$\underline{s}_{10}$
$\underline{s}_2$
$\underline{s}_9$
$\underline{s}_3$
$\underline{s}_{12}$
$\underline{s}_4$
$\underline{s}_{11}$
$\underline{s}_4$
$\underline{s}_{14}$
$\underline{s}_5$
$\underline{s}_{13}$
$\underline{s}_6$
$\underline{s}_{16}$
$\underline{s}_7$
$\underline{s}_{15}$
$M = 8$
$Q = 16$
$d_{free} = 2\sqrt{E_s}$
$N_{free} = 5$
$G = 3$ dB
8 State Code:
$m = 3$
$(\underline{s}_1/\underline{s}_9, \underline{s}_3/\underline{s}_{11}, \underline{s}_5/\underline{s}_{13}, \underline{s}_7/\underline{s}_{15})$
$(\underline{s}_2/\underline{s}_{10}, \underline{s}_4/\underline{s}_{12}, \underline{s}_6/\underline{s}_{14}, \underline{s}_8/\underline{s}_{16})$
$(\underline{s}_3/\underline{s}_{11}, \underline{s}_1/\underline{s}_9, \underline{s}_7/\underline{s}_{15}, \underline{s}_5/\underline{s}_{13})$
$(\underline{s}_4/\underline{s}_{12}, \underline{s}_2/\underline{s}_{11}, \underline{s}_8/\underline{s}_{16}, \underline{s}_6/\underline{s}_{14})$
$(\underline{s}_8/\underline{s}_{16}, \underline{s}_6/\underline{s}_{14}, \underline{s}_4/\underline{s}_{12}, \underline{s}_2/\underline{s}_{10})$
$(\underline{s}_7/\underline{s}_{15}, \underline{s}_5/\underline{s}_{13}, \underline{s}_3/\underline{s}_{11}, \underline{s}_1/\underline{s}_9)$
$(\underline{s}_6/\underline{s}_{14}, \underline{s}_8/\underline{s}_{16}, \underline{s}_2/\underline{s}_{10}, \underline{s}_4/\underline{s}_{12})$
$(\underline{s}_5/\underline{s}_{13}, \underline{s}_7/\underline{s}_{15}, \underline{s}_1/\underline{s}_9, \underline{s}_3/\underline{s}_{11})$

Figure 4. Trellis Code for 16-ary Biorthogonal Signal Set.

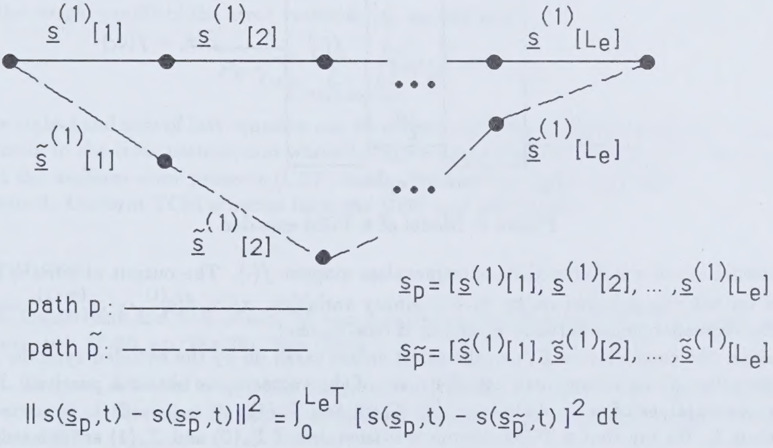

$\underline{s}^{(1)}[1]$
$\underline{s}^{(1)}[2]$
$\underline{s}^{(1)}[L_e]$
$\tilde{\underline{s}}^{(1)}[1]$
$\tilde{\underline{s}}^{(1)}[L_e]$
$\tilde{\underline{s}}^{(1)}[2]$
path p:
path $\tilde{p}$:
$\underline{s}_p = [\underline{s}^{(1)}[1], \underline{s}^{(1)}[2], \dots, \underline{s}^{(1)}[L_e]]$
$\underline{s}_{\tilde{p}} = [\tilde{\underline{s}}^{(1)}[1], \tilde{\underline{s}}^{(1)}[2], \dots, \tilde{\underline{s}}^{(1)}[L_e]]$
$\| s(\underline{s}_p, t) - s(\underline{s}_{\tilde{p}}, t) \|^2 = \int_0^{L_e T} [s(\underline{s}_p, t) - s(\underline{s}_{\tilde{p}}, t)]^2 \, dt$

Figure 5. Two Diverging Paths.

UNIFORM DISTANCE AND ERROR PROBABILITY PROPERTIES OF TCM SCHEMES[1]

EZIO BIGLIERI[2] and PETER J. McLANE[3]

[2]Electrical Engineering Department, UCLA, Los Angeles
[3]Department of Electrical Engineering, Queen's University, Kingston, Ontario

The two parameters commonly used for comparing trellis-coded modulation (TCM) schemes [3] are the free Euclidean distance and the error probability. Algorithms for their computation can be classified in two categories. The first one, based on consideration of pairs of states, is extremely general, but has a computational complexity increasing with N^2 (where N is the number of states in the trellis). The second category includes algorithms whose computational complexity increases with N, but their applicability depends on the TCM scheme. If they are applicable for the evaluation of free distance, we say that the scheme has the *Uniform Distance Property* (UDP), while if this is possible for the evaluation of an upper bound (based on union-Bhattacharyya or union-Chernoff bound) to error probability we say that it has the *Uniform Error Property* (UEP).

In this paper we derive a set of conditions that are sufficient for both UDP and UEP. Our results, which extend those of [3,4], are also applicable to the case where the TCM scheme is described by using the analytical representation proposed in [1]. Moreover, our technique can be generalized to encompass transmission over a not necessarily Gaussian memoryless channel, where the metric used for detection may not be maximum-likelihood, and hence the appropriate distance measure is not Euclidean. (In particular, a channel with phase offset or with independent, amplitude-only fading.)

We consider rate $m/(m+1)$ TCM schemes, and the model shown in Fig. 1, in which a linear convolutional encoder is followed by a memoryless mapper $f(\cdot)$. The output at time i, say x_i, depends on the values taken on by $m+1$ binary variables: $x_i = f(c_i^{(1)}, \cdots, c_i^{(m+1)}) = f(\mathbf{c}_i)$, where the correspondence between x_i and $\mathbf{c}_i$ is one–to–one.

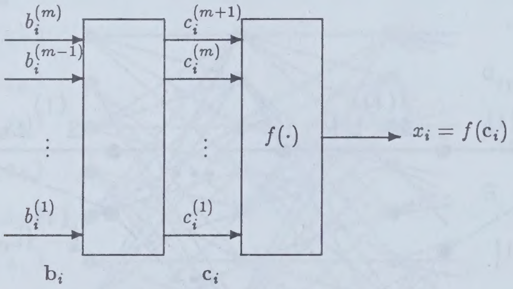


Figure 1: Model of a TCM encoder

Consider the constellation $\mathcal{X}$, i.e., the set of values taken on by the encoded symbols x_i. By fixing the value of one component, say the ν-th, of the vector $\mathbf{c}$, we obtain a partition of $\mathcal{X}$ in two subconstellations of m symbols each, say $\mathcal{X}_\nu(0)$ and $\mathcal{X}_\nu(1)$.

Definition 1. We say that a TCM scheme is *ν-isometric* if $\mathcal{X}_\nu(0)$ and $\mathcal{X}_\nu(1)$ are related by an isometry, i.e., they have the same mutual Euclidean distances between elements.

Definition 2. A TCM scheme is called *uniform* if there exists an integer ν, $0 \leq \nu \leq m+1$, such that: (a) The scheme is ν-isometric. (b) The m-tuples $\mathbf{k}_i = (c_i^{(1)}, \cdots, c_i^{(\nu-1)}, c_i^{(\nu+1)}, \cdots, c_i^{(m+1)})$

---

[1] This work was partially supported by NATO under Grant No. 0368/87 and by NSERC Canada.

are independent. (c) The sequence of m-tuples $\mathbf{k}_i$ uniquely determines the sequence of $(m+1)$-tuples $\mathbf{c}_i$.

Definition 3. A TCM scheme modeled as in Fig. 1 is called an *Ungerboeck code* if it is ν-isometric for some ν. (In actual designs, it is customary to choose $\nu = 1$).

The following result can be proved:

Theorem 1. Ungerboeck codes are uniform. □

If we denote by x the channel symbol, and by $b_1, \cdots, b_n$ the binary source symbols on which it depends, we can write $x = f(b_1, \cdots, b_n)$. [1] By assuming here that the source symbols take on values ± 1, the function $f(\cdot)$ can be expanded in the form of a finite series, namely

$$f(b_1, \cdots, b_n) = q_0 + \sum_i q_i b_i + \sum_{j>i} q_{ij} b_i b_j + + \sum_{k>j>i} q_{ijk} b_i b_j b_k + \cdots + q_{12\cdots n} b_1 b_2 \cdots b_n$$

where the q's are some constants that can be computed. Now, consider the case in which there are exactly $m+1$ terms in the finite series above [2, p.29]. Under this assumption, by identifying the terms $b_i, b_i b_j, b_i b_j b_k, \cdots$, of the finite series with the $c_i^{(\ell)}$ of our model, we have the following Theorem.

Theorem 2. An analytically described TCM scheme is uniform if it satisfies the following conditions: (a) $x_i = \sum_{\ell=1}^{m+1} \epsilon^{(\ell)} c_i^{(\ell)}$, where the terms $c_i^{(\ell)}$ are products of b_i's, and $\{\epsilon^{(\ell)}\}_{\ell=1}^{m+1}$ denotes a superincreasing sequence. (b) The TCM scheme is ν-isometric for some ν.□

Consider first the computation of the free distance. If L denotes the length of an error event, and $\mathbf{e}_k, \mathbf{e}_{k+1}, \cdots, \mathbf{e}_{k+L-1}$ is an error sequence, by defining $w^2(\mathbf{e}_i) = \min_{\mathbf{c}_i} \| f(\mathbf{c}_i) - f(\mathbf{c}_i \oplus \mathbf{e}_i) \|^2$ (the *Euclidean weight* of error vector $\mathbf{e}_i$), the TCM scheme has the *Uniform Distance Property* (UDP) if

$$d^2_{\text{free}} = \min_{L} \min_{\mathbf{e}_0, \cdots \mathbf{e}_{L-1}} \sum_{i=0}^{L-1} w^2(\mathbf{e}_i).$$

Consider next the computation of error probability for transmission over the AWGN channel with noise power spectral density $N_0/2$. If, by defining the function

$$W(Z, \mathbf{e}) = \sum_{\mathbf{c}} P[\mathbf{c}] Z^{d^2[f(\mathbf{c}), f(\mathbf{c} \oplus \mathbf{e})]}, \tag{1}$$

called the *weight profile* of the error vector $\mathbf{e}_i$ [4], we can write

$$P_E \leq \sum_{L} \sum_{\mathbf{e}_1, \cdots, \mathbf{e}_L} \prod_{i=1}^{L} W(e^{-1/4N_0}, \mathbf{e}_i), \tag{2}$$

then the right-hand side of last equation can be computed by using a state diagram whose nodes corresponds to the error vectors, and whose branches carry the labels $W(Z, \mathbf{e}_i)$. In this case, we say that the *uniform error property* (UEP) holds. We have the following result:

Theorem 3. Uniform TCM schemes have the UDP and the UEP.□

References

[1] A. R. Calderbank and J. E. Mazo, "A new description of trellis codes," *IEEE Trans. Inform. Theory,* vol. IT-30, pp. 784-791, Nov. 1984

[2] J. M. Turgeon, *Minimal complexity design of analytically described trellis codes,* MS Thesis, Department of Electrical Engineering, Queen's University at Kingston, Ontario (Canada), August 1987

[3] G. Ungerboeck, "Channel coding with multilevel-phase signals," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 55-67, Jan. 1982

[4] E. Zehavi and J. K. Wolf, "On the performance evaluation of trellis codes," *IEEE Trans. Inform. Theory,* vol.IT-33, no.2, pp.196-202, March 1987

CONSTRUCTION OF TRELLIS-DECODABLE ERROR-CORRECTING LINE CODES

MARK A. HERRO, VIVEK TELANG and A. ROBERT CALDERBANK
Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556,
Tellabs Research Center, Mishawaka, IN 46545, and
AT&T Bell Laboratories, Murray Hill, New Jersey 07974

I. INTRODUCTION

The design of balanced error-correcting codes has received a lot of attention in recent literature. Besides their error-control capability, these codes also have power spectral densities that make them attractive for use on the fiber optic channel and for data storage on magnetic tape. Since these codes are balanced, the number of 1's in every code word equals the number of 0's. This property guarantees a null at DC in the power spectral densities of these codes. In this paper we show ways of constructing error-correcting trellis-decodable balanced codes.

II. CODE CONSTRUCTION

We construct code words by a two-layered method similar to the approach described in [1] and [2]. We define a signal constellation that consists of balanced binary vectors of length 2^N and we consider a nested or multi-level partition of this signal constellation. An L-level partition of the signal constellation is denoted by a sequence of partitions, $\Gamma_1, \Gamma_2, \ldots, \Gamma_L$. The partition determines a rooted tree with $L + 1$ levels, where the root of the tree is the signal constellation Γ_0 itself. The subsets that form the partition Γ_L are the leaves of this tree. We define the subset distance δ_i by the minimum Hamming distance between symbols within a subset at the i^{th} level. It follows from the definition that $\delta_0 \leq \delta_1 \leq \cdots \leq \delta_{L-1}$. An L-level code $C = [C_1, C_2, \ldots, C_L]$ with component codes $C_1, C_2, \ldots, C_L$ is the set of all sequences $\{(a_1^k, a_2^k, \ldots, a_L^k)\}$ of subsets in Γ_L that satisfy $(a_i^k) \in C_i$ for all $i = 1, 2, \ldots, L$.

In this paper we focus on 2-level codes. As an example we consider the signal constellation Γ_0 consisting of balanced binary vectors of length 2^N with a minimum Hamming distance $\delta_0 = 2^{N-1}$. The generator matrix for these vectors consists of the 2^N-bit all 1 row vector and the vectors $(\mathbf{z}, 0)$ where $\mathbf{z}$ is a row in the generator matrix of the $(n, k, d_{\min}) = (2^N-1, N, 2^{N-1})$ simplex code. Eliminating the all 0 and the all 1 code words determines 2^N-1 pairs of complementary vectors with weight 2^{N-1}, and these form the signal constellation Γ_0. The first partition divides the signal constellation into the 2^N-1 pairs of complementary vectors. The second partition separates the two code words in each pair. The subset distances are $\delta_0 = 2^{N-1}$ and $\delta_1 = 2^N$. The 2-level code $C = [C_1, C_2]$ consists of all sequences $(a_1^k\ a_2^k)$ where $(a_1^k) \in C_1$ and $(a_2^k) \in C_2$. Note that C_1 is a "2^N-1"-ary code, whereas C_2 is a binary code. Let d_1 and d_2 be the minimum Hamming distance in C_1 and C_2 respectively. Then the minimum Hamming distance of C is given by

$$d(\mathrm{C}) = \min(d_1\delta_0, d_2\delta_1) = \min(2^{N-1}d_1, 2^N d_2). \tag{1}$$

III. PROPERTIES OF THE CODE

Since the signal constellation Γ_0 consists of balanced binary vectors, every code word is also balanced, *i.e.*, the digital sum variation (DSV) of every code word is zero. The maximum runlength $(l + 1)$ occurs when the 2^N-bit vector $00 \cdots 0011 \cdots 11$ from Γ_0, is followed by $11 \cdots 1100 \cdots 00$ or vice versa so that $l = 2^N-1$. The minimum runlength $(b + 1)$ is 1. Since every binary 2^N-tuple from Γ_0 is balanced, the DSV is zero at the boundaries between any two 2^N-tuples. The maximum accumulated charge $C = 2^{N-1}$ which occurs between the two binary vectors from Γ_0, $00 \cdots 0011 \cdots 11$ and $11 \cdots 1100 \cdots 00$. As a simple example, we can construct 2-level codes C with parameters $(b, l, C) = (0, 3, 2)$ and any desired minimum distance d(C) by choosing $N = 2$ and varying the component codes C_1 and C_2. The 2-level partition of the signal constellation is depicted in Figure 1. Choosing the component code C_1 to be the $(m, m-1, 2)$ zero-sum code, and leaving the second level uncoded $(d_2 = 1)$ results in a 2-level code C with $d_{\min} = 4$. Since the component code C_1 allows $(2^N-1)^{m-1}$ distinct code vectors and the component code C_2 generates 2^m code words from each code vector, the total number of code words is $N_{cw} = (2^N-1)^{m-1} \times 2^m$. The transmission rate R is given by

$$R = \frac{\log_2 [2^m \times (2^N-1)^{m-1}]}{2^N m} = \frac{\log_2 [2\,(2^N-1)]}{2^N} - \frac{\log_2 (2^N-1)}{2^N m}. \tag{2}$$

The asymptotic code rate is $\dfrac{\log_2 [2\,(2^N-1)]}{2^N}$. For our choice of $N = 2$, we obtain a rate 0.6462 2-level code C with parameters $(b, l, C) = (0, 3, 2)$ and $d(C) = 4$ [3]. The maximum runlength and maximum accumulated charge are lower than those in the rate 8/16 (0, 7, 5) code proposed by Ferreira [4] and the rate 9/16 code (0, 5, 5) code proposed by Blaum [5].

The multilevel approach is not restricted to signal constellations consisting of balanced vectors. In fact, higher rate codes can be constructed if each code word is not constrained to be a balanced code word. This results in a larger number of vectors in Γ_0 and consequently a higher code rate. We have used this approach to construct a 3-level code with parameters $(b, l, C) = (0, 4, 3)$, $d(C) = 4$ and asymptotic code rate 0.6643.

IV. TRELLIS DECODING

Maximum likelihood decoding of an L-level code $C = [C_1, C_2, \dots, C_L]$ combines maximum likelihood decoders for $C_1, C_2, \dots, C_L$ into a single decoding trellis. The states in the trellis are the L-tuples $[S_1, S_2, \dots, S_L]$ where S_i is a state in the maximum likelihood decoding trellis β_i for component code C_i. For the 2-level balanced code described above, the component code C_1 can be decoded by a 2^N-1-state trellis. Since the second level of the partition is uncoded, the maximum likelihood decoding of the 2-level code C can be accomplished with the 2^N-1-state trellis for C_1.

VI. REFERENCES

[1] Calderbank, A. R., "Multi-level Trellis Codes for the Gaussian Channel and for Channels subject to Impulsive Noise," *IEEE Trans. Inform. Theory*, submitted for publication.

[2] Ungerboeck, Gottfried, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55-67, Jan. 1982.

[3] Telang, Vivek and Mark A. Herro, "A Constructive Algorithm for Balanced DC-free Codes with $d_{min} = 4$," *Proc. of the 1987 Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, March 1987.

[4] Ferreira, Hendrik C., "Lower bounds on the minimum Hamming distance achievable with runlength constrained or dc-free block codes and the synthesis of a (16,8) $d_{min} = 4$ dc-free block code," *IEEE Trans. on Magn.*, vol. MAG-20, pp. 881-3, September 1984.

[5] Blaum, Mario, "Error-correcting d.c.-free block codes," *IEEE Trans. Inform. Theory*, (submitted for publication).

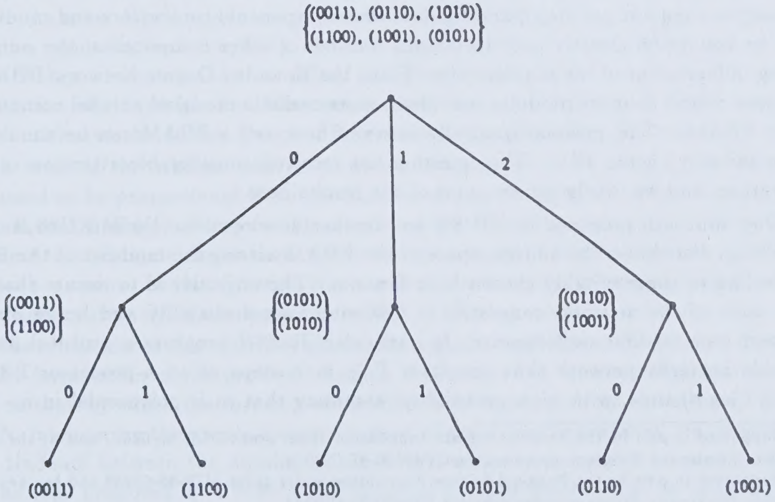


Figure 1. The 2-level partition of the six balanced binary 4-tuples.

Deterministic Simulations of PRAMs on Bounded Degree Networks

(Extended Abstract)

Kieran T. Herley [1] and Gianfranco Bilardi [2]
Computer Science Department, Cornell University, Ithaca, NY14853

Abstract

In this paper we consider the problem of simulating a PRAM with n processors and memory size $m \geq n$ on an n-node bounded degree network. We present a scheme which simulates an arbitrary PRAM step in $O((\log n \log m)/\log\log n)$ time in the worst case on an expander-based network. For a wide range of m this time bound matches a previously established lower bound.

1 Introduction

Parallel Random Access Machines(PRAMs) play a central role in the study of parallel computation and in the development of parallel algorithms[KR88]. A PRAM is essentially a set of synchronous processors connected to a shared memory. The basic feature of the PRAM is that memory references to distinct memory cells can be made simultaneously by different processors in the same basic step. Simultaneous references to the same memory cell are allowed in various degrees in different variants of the PRAM model. By abstracting several of the constraints that arise in the physical realization of a multiprocessor, the PRAM model provides an attractive framework for algorithm design. However, performance estimates obtained for the PRAM are not necessarily accurate when referred to a realistic multiprocessor [Sny86].

Two important constraints not captured by PRAMs are the granularity of the memory (the memory is partitioned into banks or modules each of which can respond to only one request per step), and the fact that components (processors and modules) can be connected directly only to a small number of other components, the number being independent of the machine size. Thus, the Bounded Degree Network(BDN) of processors and memory modules provides a more realistic model of parallel computers than PRAMs. The question naturally arises of how well a PRAM can be simulated by a suitably chosen BDN. This question has received considerable attention in the literature, and we briefly review some of the results here.

One approach proposed in [MV84] and further developed in [Upf84,KU86,Ran87, LPP87a], distributes the address space of the PRAM among the modules of the BDN according to some suitably chosen hash function. The objective is to ensure that, at any node of the network, congestion is low with high probability and hence obtain a good *expected-time* performance. In particular [Ran87] proposes a protocol for an n-node butterfly network that simulates $T \geq m/n$ steps of an n-processor PRAM in $O(T \log n)$ time, with high probability, assuming that m is polynomial in n. The

---

[1] Supported in part by the National Science Foundation under grant DCR-86-02307 and by the Joint Services Electronics Program under contract F49620-87-C-0044.

[2] Supported in part by the National Science Foundation under grant MIP-86-02256 and by the Joint Services Electronics Program under contract F49620-87-C-0044.

protocol is deterministic, except for the random choice of the hash function that defines the address map.

Another approach, also proposed in [MV84], relies on representing the contents of a PRAM memory cell by using several BDN cells. This approach forms the basis for a number of deterministic simulation schemes [UW87,AHMP87,LPP87b] including those proposed in this paper. The goal of these schemes is to provide good *worst-case* performance.

In [UW87], a scheme is advanced to simulate an arbitrary step of a PRAM with n processors and $m \geq n$ memory cells by an n node BDN in time $O(\log n \log m \log \log m)$. Nodes of the BDN consist of a RAM processor and a memory module and have an interconnection with the structure of the AKS sorting network[AKS83,Lei85]. This time bound was subsequently improved to $O(\log n \log m)$ in [AHMP87], and is further improved to $O((\log n \log m)/ \log \log n)$ in the present paper. Our network is also an n-node BDN, with a particular structure to be described later.

An $\Omega((\log n \log m)/ \log \log m)$ lower bound for the time to simulate a step of a PRAM with n processors and $m \geq n$ memory cells by and n-node BDN was established independently in [AHMP87] and [KU86]. The lower bound makes the so-called point-to-point assumption, that if different copies of a variable are generated by a given BDN processor, then they must be transmitted by the network as distinct entities. Furthermore it is assumed that, for some $\epsilon > 0$, $\Omega(n^{2+\epsilon}) \leq m \leq 2^{O(\sqrt{n \log n})}$. By refining the proof technique of [AHMP87] and [KU86], we can extend the lower part of the range for m to $m = \Omega(n^{1+\epsilon})$.

It is interesting to observe that the simulation described in this paper complies with the point-to-point assumption, and that its performance matches the cited lower bound as long as $\Omega(n^{1+\epsilon}) \leq m \leq O(2^{(\log n)^\alpha})$, for some $\epsilon > 0$ and some $\alpha > 0$.

In [LPP87b] a PRAM simulation is presented on a BDN consisting of an $n \times n$ mesh-of-trees [Lei81,NMB83], where the roots of the trees are fully fledged RAM processors, while the other $\Theta(n^2)$ nodes are packet-switching elements. The scheme is attractive for the regularity and simplicity of the interconnection. For m polynomial in n, a PRAM step is simulated in time $O(\log^2 n/ \log \log n)$, the same as that achieved by our simulation for this range of m. However, it should be noted that the use of $\Theta(n^2)$ nodes, albeit mere switchs, makes the cited lower bound inapplicable to the mesh-of-trees.

In most of the studies referred to above the number n of nodes in the BDN is assumed to be proportional to to the number of processors in the PRAM being simulated, and is independent of of the size m of the PRAM memory. It is certainly of interest to investigate simulations where the the size of the BDN is allowed to depend on both n and m. In this direction it is worth mentioning the scheme of [Vis84], which has running time $O(\log n + \log m)$ on a BDN of $O(n \log n + m \log m)$ nodes.

The remainder of this extended abstract is organized as follows. Section 2 is devoted to the formulation of the problem. Section 3 discusses the memory organization, i.e. the method by which the memory of the PRAM is represented by the memory of the BDN. Introducing the graph-theoretic notion of generalized expanders, we investigate the trade-off between the number of copies and memory congestion in organizations of the type proposed in [UW87]. The choice of an appropriate point in this trade-off is critical is order to balance the work needed to extract the relevant information from the memory modules with the work needed to route this information to the intended processors. Section 4 describes the memory access protocol. In several steps of the protocol, the necessity arises to process sets of elements distributed unevenly among the nodes of the BDN. This problem is solved by a careful combination and adaptation of sorting [AKS83,Lei85] and routing [PU86] techniques. Finally an analysis of the protocol yields the bounds stated above.

2 Problem Formulation

We consider a PRAM of n processors, $P_1^{PRAM}, P_2^{PRAM}, \ldots, P_n^{PRAM}$ sharing a set U of m memory cells. Our goal is to simulate such a PRAM by a BDN of n nodes $(P_1, M_1), (P_2, M_2), \ldots, (P_n, M_n)$ where P_i is a RAM processor and M_i is a memory module. We are interested in simulation algorithms that are deterministic and on-line, i.e. the simulation of a step is independent of future steps.

We assume that the instruction to be executed by PRAM processor P_i^{PRAM} at a given step is available to the corresponding BDN processor P_i at the beginning of the simulation of this step. Since simulation of arithmetic and logical operations is straightforward, we can regard a generic PRAM step simply as an n-tuple of memory accesses. A memory access can either be of the form read(u), ($u \in U$), which results in the contents of the memory cell u being copied into a distinguished register of the executing processor, or of the form write(u,x), ($u \in U, x$ an integer value), which results in the integer value x being written to memory cell u. For simplicity we assume that all the accesses in a given step are of the same type (read or write), and that no two accesses refer to the same memory cell. Both assumptions can be removed by established techniques [Vis83] without altering the stated time bounds, as already noted in [UW87].

In any simulation scheme, upon termination of the simulation of a given PRAM step, the state of the PRAM memory must be uniquely reconstructible from the state of the BDN memory. The method chosen for representing the state of the PRAM memory by the state of the BDN memory is referred to as the *memory organization.* The algorithm that probes the state of the BDN memory when simulating a read step, and that modifies it when simulating a write step is called the *memory access protocol.*

A simulation scheme is consistent if after the simulation of an arbitrary sequence of PRAM steps, the simulation of a step where processor P_i^{PRAM} executes read(u), results in loading the distinguished register of P_i with the correct value of u, which is if course the value assigned to u by the the latest instruction of the form write(u,x).

In the next two sections we describe and analyse a new simulation scheme. Section 3 is devoted to the memory organization, and Section 4 to the memory access protocol.

3 Memory Organization

Following [UW87], we consider a memory organization where, for each PRAM variable $u \in U$, the BDN maintains $2c - 1$ copies in the memory modules of a set

$\Gamma(u) = \{\gamma_1(u), \gamma_2(u), \ldots, \gamma_{2c-1}(u)\}$ of distinct nodes. Each copy contains a value and a timestamp indicating the time at which that copy was last written. When simulating a step where u is written, at least c copies of u are updated. When simulating a step where u is read, at least c copies of u are retrieved: the one with the most recent timestamp is guaranteed to contain the correct value of u. We will adopt the same read/write discipline.

This type of memory organization can be modelled conveniently as a bipartite graph $G = (U, V, E)$ where U is the set of PRAM variables, V is the set of the nodes of the BDN, and an edge $(u, \gamma(u))$ represents the fact that the node $\gamma(u)$ contains a copy of the variable u.

Let $S \subseteq U$ and let $F \subseteq E$ be a set of edges incident on S with exactly k edges incident upon each $u \in S$. We say that the subset of V on which the edges of F are incident is a *k-neighbourhood* of S and denote it by $\Gamma_F(S)$.

When the copy scheme described above is adopted, the simulation of a PRAM step accessing the variables of S requires that the BDN access all the copies of some *c-neighbourhood*, say $\Gamma_F(S)$. Clearly this takes at least $c|S|/|\Gamma_F(S)|$ time, since at least $c|S|$ copies have to be accessed from $|\Gamma_F(S)|$ modules. It is desirable to have a memory organization with large c-neighbourhoods. The preceding remarks motivate the following definition.

Definition 1 *A bipartite graph* $G = (U, V, E)$ *with* $|U| = m$ *and* $|V| = n$, *and with each node in* U *having degree d is a* (λ, d, c, σ)*-generalized expander if, for every* $S \subseteq U$ *such that* $|S| \leq \sigma n$ *and for every c-neighbourhood* $\Gamma_F(S)$ *of* S, $|\Gamma_F(S)| \geq \lambda c |S|$.

Clearly $\lambda \leq 1$ and $\sigma \leq 1/\lambda c$ are necessary conditions for the existence of a (λ, d, c, σ)-generalized expander. By counting arguments, we can establish the existence of certain types of generalized expanders.

Theorem 1 *For every* n, m, g *and* λ, *with* $m \geq n$, $g \geq 1$ *and* $\lambda < 1$, *there is a* γ *depending only on* λ *such that for every* $c \geq \gamma \log(m/n)/\log g$ *there exists a* $(\lambda, 2c-1, c, 1/gc)$*-generalized expander* $G = (U, V, E)$ *with* $|U| = m$ *and* $|V| = n$.

Upfal and Wigderson[UW87] established the existence of a family of $(\lambda, 2c-1, c, 1/(2c-1))$-generalized expanders for every $\lambda \leq 1/2$, with $c \geq \gamma \log m$. These expanders and similar ones define the memory organizations used in [UW87,AHMP87,LPP87b]. They can essentially be derived from Theorem 1 by choosing $g = 2$.

The interest of Theorem 1 lies in the introduction of the parameter g, which implies a trade-off between the number of copies and the largest set for which expansion can be guaranteed. Roughly speaking, increasing g reduces the number of copies to be processed, but creates higher congestion at memory modules, thereby making the job of accessing those copies more difficult. As a first step toward a quantitative analysis of this trade-off we need to derive some properties of generalized expanders. The following lemma is essentially a rephrasing of the definition of generalized expanders.

Lemma 1 *Let* $G = (U, V, E)$ *be a* $(\lambda, 2c-1, c, 1/gc)$*-generalized expander. If* $S \subseteq U$ *and* $F \subseteq E$ *defines a c-neighbourhood* $\Gamma_F(S)$ *of size smaller than* $\lambda n/g$, *then* $|S| \leq |\Gamma_F(S)|/\lambda c$.

Consider now a memory organization represented by a bipartite graph $G = (U, V, E)$ and let $S \subseteq U$ be a set of PRAM variables to be accessed. The copies of the variables in S will be distributed among the BDN nodes with node $v \in V$ containing $s_v = |\{(u, v) : u \in S, (u, v) \in E\}|$ copies. We are interested in analysing the effect of accessing $\min(s_v, q)$ copies from each node $v \in V$ as a function of the integer q. This operation, which we denote by *Decimate(S,q)*, will in fact form the basis of the memory access protocol to be described in the next section.

The variables of S can be partitioned into two sets: those for which at least c copies have been accessed, which we will call the *dead* variables, and the remaining ones which we call the *live* variables. Dead variables play no further role in the simulation since enough copies have been accessed to effect either a read or write. For the live variables, more copies need to be accessed. Therefore, we are interested in bounding the size of the set of variables left alive after *Decimate(S,q)*.

Formally, the set of copies accessed by *Decimate(S,q)* is represented by a set of edges $E' \subseteq E$ such that if $(u, v) \in E'$ then $u \in S$, and for each $v \in V$, there are exactly $\min(s_v, q)$ edges in E' incident upon v. Then, L is the subset of the nodes of S with degree at least c in the graph $(U, V, E - E')$.

Lemma 2 *With the above notation, if the memory organization is a* $(\lambda, 2c - 1, c, 1/gc)$*-generalized expander* $G = (U, V, E)$ *and if* $q > 2\lceil (cg/n)|S| \rceil / \lambda$ *then the set* L *of variables remaining alive after* Decimate(S,q) *satisfies the relation*

$$|L| \leq \left[\frac{2(1 - \lambda)}{q - 2\lceil cg|S|/n \rceil + 1} \right] |S|.$$

We shall call the quantity $|L|/|S|$ the *survival factor* of the operation *Decimate(S,q)*. *Decimate* is the building block of the memory access protocol described in the next section. In contrast, the protocols of [UW87] and [AHMP87] are built around a halving procedure which although implemented differently, has the same effect as a decimation with survival factor 1/2.

Intuitively, the advantage of decimation over halving can be understood as follows. From standard considerations on the diameter of an n-node BDN it is clear that any implementation of halving will require $\Omega(\log n)$ time in the worst case. However it will be shown in the next section that $O(\log n)$ time is essentially sufficient for the execution of a *Decimate(S,O*$(\log n)$*)*, as long as $|S| \leq n/(2c - 1)$. From Lemma 2 we can see that, for $|S| \leq n/(2c - 1)$, choosing $g = \log n$ and q a suitable multiple of g yields a survival factor of $O(1/\log n)$. In conclusion, decimations can kill more variables than halving in the same amount of time. This observation is also employed in the scheme of [LPP87b].

4 The Memory Access Protocol

In this section we describe a memory access protocol for the simulation of one step of a PRAM with n processors and m memory cells. We assume that the memory organization is a $(\lambda, 2c - 1, c, 1/gc)$-generalized expander with $c = \gamma \log(m/n)/\log g$. If is further assumed that each node of the BDN stores in its local memory a table giving for each PRAM memory cell u the nodes $\gamma_1(u), \gamma_2(u), \ldots, \gamma_{2c-1}(u)$ containing copies of u.

Let X be the set of PRAM variables to be accessed in a given PRAM step. The goal of the simulation is to access a majority of the copies of each variable in X. We shall describe the protocol for simulating a read step, the protocol for a write step being very similar. A high level description of the protocol follows.

I For each processor P_i, generate u_i, the name of the variable it wishes to read.

II For each u_i and each $j \in \{1, 2, \ldots, 2c-1\}$ generate a *request packet* containing an origin P_i, a destination $\gamma_j(u_i)$, the label u_i itself, and a field to store the value and the timestamp of the copy of u_i contained in node $\gamma_j(u_i)$.

III Deliver at least c request packets from each origin to their destinations, and return these packets to their origins with the values and the timestamps.

IV At P_i, select the value from the returned packet(s) with the most recent timestamp as the result of read(u_i).

Steps I, II and IV are straightforward and are accomplished in time $O(1)$, $O(c)$, and $O(c)$ respectively. We focus our attention on Step III. The latter will be implemented as a sequence of decimations, the first one applied to the set X, and the remaining decimations each applied to the set of variables left alive by the previous decimation.

More specifically, we shall define a procedure *Decimate(S,q)* with the following properties: (i) S is a variable parameter and represents a subset of U of size at most n distributed among the nodes of the BDN, with at most element one per node; (ii) the second argument q is a value parameter of type integer; (iii) the execution of *Decimate(S,q)* will cause $\min(s_v, q)$ packets destined for node v to reach that node and to be returned to their respective origins, with the appropriate values and timestamps (recall from Section 3 that s_v denotes the number of copies of variables in S contained in node v); (iv) the value of parameter S is modified by the execution of *Decimate(S,q)* so that, upon termination, S represents the set of live variables, that is those whose origins have received fewer than c packets.

In terms of procedure *Decimate*, Step III of the memory access protocol can be written as follows.

```
        $S \leftarrow X$;
        for $h \leftarrow 1$ to $H$ do
            Decimate($S, q_h$)
```

The values of $q_1, q_2, \ldots, q_H$ and H will be specified later so that after the H^{th} iteration S is empty, i.e. all the variables in X have been accessed.

To complete the definition of the memory access protocol, it remains to specify an implementation for the procedure *Decimate.* The next subsection is devoted to a description to the operation of *balancing* and sorting which will serve as a routing primitive. The implementation of *Decimate* is given in the subsequent subsection. The final subsection provides an analysis of the running time for this implementation.

4.1 Balancing and Sorting

Let $\mathbf{k} = (k_1, k_2, \ldots, k_n)$. We say that a set Z is $\mathbf{k}$-distributed if the elements of Z are distributed among the n nodes of a BDN with k_i elements at node i. A $\mathbf{k}$-distributed set is *balanced* if $\max\{k_1, \ldots, k_n\} = \lceil |Z|/n \rceil$.

Peleg and Upfal[PU86] exhibit an algorithm for balancing a set by redistributing its elements among the nodes. Their algorithm works on a BDN $(V, E_{AKSL} \bigcup E_{PU})$, where (V, E_{AKSL}) is the graph underlying Leighton's modification of the AKS sorting network[AKS83,Lei85], and (V, E_{PU}) is a square-root expander graph[PU86,LPS86]. The result of [PU86] is stated in a model where only communication steps are counted, and where the set to be balanced has at most n elements. Restated in our model and generalized to sets of arbitrary size, the result becomes the following one.

Lemma 3 *A $(k_1, \ldots, k_n)$-distributed set Z can be balanced on the BDN $(V, E_{AKSL} \bigcup E_{PU})$ of n nodes in time $O(\log n \log K + K + \lceil |Z|/n \rceil \log n)$ where $K = max\{k_1, \ldots, k_n\}$.*

Let Z be a $(k_1, \ldots, k_n)$-distributed set and let Z' be a $(k'_1, \ldots, k'_n)$-distributed set. We say that Z *dominates* Z' if $k_i \geq k'_i$ for $i = 1, 2, \ldots, n$. It turns out that some of the information computed by the Peleg and Upfal algorithm to balance Z can be reused to balance Z'. Exploiting this fact, we obtain the following lemma, which we state for the restricted case $|Z| \leq n$, the only case that interests us here.

Lemma 4 *Let Z, $|Z| \leq n$, be a $(k_1, \ldots, k_n)$-distributed set with $max\{k_1, \ldots, k_n\} = K$. Having balanced Z, any set Z' dominated by Z can be balanced at an extra cost of $O(\log n + K)$ time.*

Various stages of the implementation of *Decimate* will consist of sorting a distributed set Z, that is rearranging the elements of Z so that the $\lceil |Z|/n \rceil$ smallest elements are at node 1, the next $\lceil |Z|/n \rceil$ are at node 2 and so on. Sorting can be accomplishd by first balancing Z, and then using the AKSL sorting algorithm[AKS83,Lei85] modified according to the techniques of [BS78] to handle the case $|Z| > n$. Sorting Z after balancing will take $O(\lceil |Z|/n \rceil \log(|Z| + n))$ time. For ease of reference we can summarize Lemmas 3 and 4, and the preceding discussion as follows. We assume $|Z|$ is polynomial in n, since this simplifies some expressions and is not restrictive in our applications.

Lemma 5 *Let Z, $|Z| = n^{O(1)}$, be a $(k_1, \ldots, k_n)$-distributed set with $max\{k_1, \ldots, k_n\} = K$. Z can be sorted on the BDN $(V, E_{AKSL} \bigcup E_{PU})$ in $O(\log n \log K + K + \log n \lceil |Z|/n \rceil)$ time. Furthermore if $|Z| \leq n$ and Z' is dominated by Z, then sorting Z' can be done at an extra cost of $O(\log n + K)$ time, having sorted Z.*

4.2 Implementation of *Decimate*

We outline an implementation of *Decimate(S,q)* on the BDN $(V, E_{AKSL} \bigcup E_{PU} \bigcup E_{Tree})$ where E_{AKSL} and E_{PU} are as defined in the previous section and (V, E_{Tree}) is a balanced binary tree. Recall that S is a set of PRAM variables, at most one per processor. A processor holding a variable of S has formed $2c-1$ request packets, one for each copy of that variable, with the format described in Step II of the memory access protocol.

Decimate(S,q)

1. Sort the request packets by destination.
2. For each destination v, determine s_v. Select $\min(s_v, q) = d_v$ request packets from those destined for node v.
3. Sort the selected requests by destination.
4. For each v generate a packet containing d_v and labelled v. Sort these packets by label.
5. For each node v generate d_v dummy tokens, each labelled v.
6. Sort the dummy tokens by label.
7. Match each request with a dummy which originated at its destination.
8. Route the request packets to their destinations by retracing the steps taken by the corresponding dummy during step 6.
9. Load values and timestamps into request packets.
10. Route the selected requests (now satisfied) back to their origins tracing the paths taken in steps 1, 3 and 6 backwards.
11. Let each origin count the number of packets received during Step 10 and if they are at least c in number, retain the value of the packet(s) with the most recent timestamp and remove the variable from S.

Lemma 6 Decimate(S,q) *can be implemented on the BDN* $(V, E_{AKSL} \cup E_{PU} \cup E_{Tree})$ *with running time* $O(\log n \log(cg) + c + q + \log n \lceil c|S|/n \rceil)$.

Proof(Sketch) Steps 1, 3 and 4 consist of sorting a distributed set of size no larger than $(2c-1)|S|$ and with at most $2c-1$ elements per node. By Lemma 5, these take $O(\log n \log c + c + \log n \lceil c|S|/n \rceil)$ time. Steps 6 and 8 consist of sorting a distributed set of size no larger than $(2c-1)|S|$ an with at most q elements per node. By Lemma 5, these steps take $O(\log n \log q + q + \log n \lceil c|S|/n \rceil)$ time. Step 10 is a combination of Steps 1, 3 and 6 and has the same running time.

Step 2 can be reduced to prefix computations [KRS85] on sequences of length $(2c-1)$ $|S|$ which can be executed on (V, E_{Tree}) in $O(\log n + \lceil c|S|/n \rceil)$ time.

Finally Steps 5, 7, 9 and 11 are sequential computations local to a given node and take $O(q)$, $O(1)$, $O(q)$, and $O(c)$ time respectively.

By summing the contributions of all the steps it is easy to see that the stated running time is achieved. □

4.3 Simulation Time

We now turn our attention to the running time of Step III of the memory access protocol consisting of a sequence of H decimations with parameters $q_1, q_2, \ldots, q_H$. Let L_i be the set of live variables after the i^{th} decimation.

For the first decimation, applied to the set X of PRAM variables, we choose $q_1 = \max(2/\lambda, 4)cg$ By Lemma 2, $|L_1| < (1-\lambda)n/(cg)$. By Lemma 6 , *Decimate(S,*q_1*)* runs in $O(\log n \log(cg) + cg + c \log n)$ time.

For the remaining decimations we choose $q_2 = q_3 = \ldots = q_H = 2g+1$. By Lemma 2, for $i \geq 2$, $|L_i| \leq (1-\lambda)|L_{i-1}|/g$. Thus, for $i \geq 2$, $|L_i| < (1-\lambda)^{i-1}|L_1|/g^{i-1} \leq (1-\lambda)^i n/g^i$. Choosing $H = \lceil \log n / \log(g/(1-\lambda)) \rceil$ ensures that $L_H = \emptyset$.

To analyse running time, let us observe that, for each $i \geq 2$, the set of requests at Step 1 and the set of dummies at Step 6 in the i^{th} decimation are dominated by the corresponding sets in the $(i-1)^{st}$ decimation. Furthermore, all these sets have size smaller than n. Therefore, by Lemma 6, *Decimate(*L_2, q_2*)* runs in time $O(\log n \log(cg) + c + g + \log n)$. By Lemma 5, each of the remaining $H-2$ decimations will take $O(c + g + \log n)$ time.

The total time for Step III, and indeed for the entire simulation is

$$O(\log n \log(cg) + cg + c \log n + (c + g + \log n)(\log n / \log g)).$$

Choosing $g = \log n$ and recalling that $c = \gamma \log m / \log g$, we arrive at the following result.

Theorem 2 *An arbitrary step of a PRAM with n processors and* $m \geq n$ *memory cells can be simulated by the BDN* $(V, E_{AKSL} \cup E_{PU} \cup E_{Tree})$ *in time* $O((\log n \log m)/\log\log n)$ *time.*

As already noted in the Introduction, our simulation scheme attains the point-to-point lower-bounds[KU86,AHMP87] in a wide range of m. The major unresolved question remaining is the explicit construction of generalized expanders for which the address map is defined by an easily computatable function rather than by a huge table stored in local memory.

References

[AHMP87] H. Alt, T. Hagerup, K. Mehlhorn, and F. P. Preparata. Simulation of idealized parallel computers on more realistic ones. *SIAM Journal of Computing*, 16(5):808–835, Nov 1987.

[AKS83] M. Ajtai, J. Komlos, and E. Szemeredi. An $O(\log n)$ sorting network. In *Proceedings of the 15th Annual Symposium on the Theory of Computing*, Boston, Massachusetts, pages 1–9, Apr 1983.

[BS78] G. Baudet and D. Stevenson. Optimal sorting algorithms for parallel computers. *IEEE Transactions on Computers*, c-27(1):84–87, Jan 1978.

[KR88] R. M. Karp and V. Ramachandran. A survey of parallel algorithms for shared-memory machines. Technical Report UCB/CSD 88/408, Computer Science Department, University of California at Berkeley, Mar 1988.

[KRS85] C.P. Kruskal, L. Rudolph, and M. Snir. The power of parallel prefix. *IEEE Transactions on Computers*, c-34(10):965–968, Oct 1985.

[KU86] A. Karlin and E. Upfal. Parallel hashing - an efficient implementation of shared memory. In *Proceedings of the 18th Annual Symposium on the Theory of Computing*, Berkeley, California, pages 160–168, May 1986.

[Lei81] F. T. Leighton. New lower bound technques for VLSI. In *Proceedings of the 22nd Symposium on the Foundations of Computer Science,*, Nashville, Tennessee, pages 1–12, Oct 1981.

[Lei85] F. T. Leighton. Tight bounds on the complexity of parallel sorting. *IEEE Transactions on Computers*, c-34(4):344–354, April 1985.

[LPP87a] F. Luccio, A. Pietracaprina, and G. Pucci. An efficient probabalistic simulations of PRAMs in VLSI. Technical Report S-87-8, Dipartmento di Informatica, Universita di Pisa, April 1987.

[LPP87b] F. Luccio, A. Pietracaprina, and G. Pucci. A new scheme for deterministic simulations of PRAMs in VLSI. Technical Report S-87-11, Dipartmento di Informatica, Universita di Pisa, June 1987.

[LPS86] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan conjecture and explicit constructions of expanders. In *Proceedings of the 18th Annual Symposium on the Theory of Computing*, Berkeley, California, pages 240–246, May 1986.

[MV84] K. Mehlhorn and U. Vishkin. Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories. *Acta Informatica*, 21(4):339–374, Nov 1984.

[NMB83] D.D. Nath, S. N. Maheshwari, and P.C.P. Bhatt. Efficient VLSI networks for parallel processing based on orthogonal trees. *IEEE Transactions on Computers*, c-32(6):569–581, June 1983.

[PU86] D. Peleg and E. Upfal. The token distribution problem. In *Proceedings of the 27th Annual Symposium on the Foundations of Computer Science*, Toronto, Canada, pages 185–194, Oct 1986.

[Ran87] A.G. Ranade. How to emulate shared memory. In *Proceedings of the 28th Annual Symposium on the Foundations of, Computer Science*, Los Angeles, California, pages 185–192, Oct 1987.

[Sny86] L. Snyder. Type architectures, shared memory and the corollary of modest potential. Technical Report TR86-03-04, Department of Computer Science, University of Washington, Mar 1986.

[Upf84] E. Upfal. A probabalistic relation between desirable and feasible models of parallel computation. In *Proceedings of the 16th Annual ACM Symposium on the Theory of Computing*, Washington, D.C., pages 258–265, May 1984.

[UW87] E. Upfal and A. Wigderson. How to share memory in a distributed system. *Journal of the ACM*, 34(1):116–127, Jan 1987.

[Vis83] U. Vishkin. Implementation of simultaneous memory address access in models which forbid it. *Journal of Algorithms*, 4(1):45–50, 1983.

[Vis84] U. Vishkin. A parallel-design distrubuted-implementation(PDDI) general-purpose computer. *Theoretical Computer Science*, 32(1):157–172, 1984.

MAPPING PYRAMID ALGORITHMS INTO HYPERCUBES

TEN-HWANG LAI AND WILLIAM WHITE
Department of Computer and Information Science
The Ohio State University - Columbus, Ohio 43210

Abstract: Embeddings of simple network structures (e.g., rectangular meshes and binary trees) into the hypercube network have served to illustrate the versatility of the hypercube; algorithms that are particularly well-suited to these structures may be efficiently modified for implementation on the hypercube. In this paper, we present two algorithms to efficiently embed the pyramid network structure into the hypercube. Each algorithm maintains low values for particular cost gauges, keeping the number of wasted hypercube processors at a minimum without unduly distorting the structure of the embedded pyramid. The efficient embedding of the pyramid into the hypercube further enhances the latter's reputation for versatility.

1 INTRODUCTION

The hypercube network has achieved a marked popularity in the field of parallel computing. The Caltech Cosmic Cube project [18] demonstrated many of the practical advantages of implementing a hypercube network. Hastad, et al., [9] and Becker and Simon [1] have examined the pronounced fault tolerance displayed by the hypercube. The network's flexibility and efficiency are also reflected in its simple expandibility, as well as a high degree of scalability [8]. The hypercube permits fast communication at a reasonable cost.

Several data transfer algorithms have been devised specifically for the hypercube by Saad and Schultz [16] and Ho and Johnsson [10], but the hypercube has also exhibited a strong propensity for adapting standard algorithms that were designed for more classical data structures and sequential architectures. Such adaptations permit the solutions of substantial problems to make good use of the hypercube's inherent advantages.

One means by which algorithms designed for other topological structures can be implemented on the hypercube is by graphically embedding the other network architecture into the hypercube. When such embeddings are efficiently accomplished (i.e., without wasting too many hypercube processors and without seriously modifying the structure of the network to make it "fit" in the hypercube), then the hypercube can effectively implement algorithms that are particularly suited for another network. Berman and Snyder [2] have examined the general problem of mapping a parallel algorithm with one communication structure into a parallel architecture with a different configuration. The problem of determining the existence of an adjacency-preserving embedding of a given graph into the hypercube has been shown to be NP-complete by Cybenko, et al., in [7].

Efficient embeddings into the hypercube have been developed for several simple configurations. Rectangular meshes have been embedded by Chan and Saad [5], Ma [13], and Ho and Johnsson [11]. Algorithms for embedding binary trees into hypercubes have been developed by Wu [21], Bhatt and Ipsen [3], and Bhatt, et al. [4]. These embeddings take advantage of the symmetric properties of the embedded networks and the hypercube.

In this paper, we present two algorithms to efficiently embed a pyramid structure in the hypercube. Several advanced algorithms that naturally lend themselves to the pyramid structure are examined by Stout [20], Miller and Stout [14], and Chang, et al. [6]. These algorithms can be effectively implemented on the hypercube by means of our embeddings.

In [19], Stout mentions a substantially different mapping of the vertices of a pyramid into a hypercube so that both expansion and dilation are optimized. The congestion associated with this mapping cannot be ascertained, however, since the images of edges are not specified. The principal contribution of this paper is a detailed mathematical analysis of our embedding algorithms. In particular, we thoroughly examine the congestion associated with our embeddings, since this measurement is an extremely important gauge of efficiency.

The comparative complexity of the pyramid structure results in an evaluation of the efficiency of our embeddings that is more complicated than that required for the simpler embeddings of trees and meshes. The fact that very efficient embeddings do exist for this more sophisticated configuration serves to enhance the reputation of the hypercube as a powerful and versatile network.

Section 2 presents basic definitions and the measurements by which we will gauge the efficiency of an embedding. Sections 3 and 4 present embeddings that yield optimal values for some of those efficiency gauges and near-optimal values for the rest.

2 DEFINITIONS AND BASIC RESULTS

Define M_k to be a $2^k \times 2^k$ mesh with vertices $V(M_k) = \{(x_1, x_2) : 0 \leq x_1, x_2 \leq 2^k - 1\}$ and edges $E(M_k) = \{(u, u') : u = (x_1, x_2), u' = (x'_1, x'_2) \in V(M_k), |x_1 - x'_1| + |x_2 - x'_2| = 1\}$.

Define P_k to be the pyramid structure with $V(P_k) = \cup_{i=0}^{k}\{(i, x_1, x_2) : (x_1, x_2) \in V(M_i)\}$ and $E(P_k) = \cup_{i=1}^{k}\{((i, x_1, x_2), (i, x'_1, x'_2)) : ((x_1, x_2), (x'_1, x'_2)) \in E(M_i)\} \cup \cup_{i=1}^{k}\{((i, x_1, x_2), (i - 1, \lfloor \frac{x_1}{2} \rfloor, \lfloor \frac{x_2}{2} \rfloor)) : (x_1, x_2) \in V(M_i)\}$.

Intuitively, P_k contains M_0 through M_k, with each vertex in M_i having four offspring in M_{i+1}. $V(P_k)$ can be separated into *levels*; specifically, $(i, x_1, x_2) \in V(P_k)$ is on level i. The subgraph generated by the level k vertices of P_k is called the *base* of the pyramid. The vertex on level 0 of P_k is called the *apex* of the pyramid. P_k is said to have *height* k. P_1 is illustrated in Figure 1.

Note that $|V(P_k)| = \frac{1}{3}(4^{k+1} - 1)$ and $|E(P_k)| = 4^{k+1} - 2^{k+2}$.

Define H_n to be the hypercube with $V(H_n) = \{0, 1\}^n$ and $E(H_n) = \{(x, y) : x, y \in V(H_n)$ and x and y differ in exactly one bit$\}$. For $x \in V(H_n)$, if $1 \leq i \leq n$, let x^i denote the element of $V(H_n)$ differing from x in the i-th bit (counting from the right). Similarly, $x^{i,j}$ differs from x in the i-th and j-th bits, and so on.

For $x \in V(H_n)$ and $b \in \{0, 1\}$, let bx denote the element of $V(H_{n+1})$ that has $(n+1)$-st bit value b and bit values identical to x in bits 1 through n. More generally, for $b_1, b_2, \ldots, b_{t-1}, b_t \in \{0, 1\}$, $b_t b_{t-1} \ldots b_2 b_1 x$ denotes the element of $V(H_{n+t})$ with $(n + i)$-th bit value b_i and bit values identical to x in bits 1 through n.

An *embedding* h of a graph G into a graph G' is a map of the vertices of G into the vertices of G' in a one-to-one fashion, combined with a map of the edges of G into the simple paths of G' so that if $e = (u, v) \in E(G)$, then $h(e)$ is a simple path of G' with endpoints $h(u)$ and $h(v)$.

We now define three gauges of the cost of a graph embedding $h : G \rightarrow G'$. The ***expansion*** of h is the ratio of the size of $V(G')$ to the size of $V(G)$, i.e., $|V(G')|/|V(G)|$. The expansion gauges how much of G' is not directly used in the embedding of G; the closer the expansion value is to one, the more efficient the embedding's utilization of G.

For $e \in E(G)$, the *dilation* of e under embedding $h : G \rightarrow G'$ is the length of the path $h(e)$ in G'. The *maximum dilation* of h, denoted $Dil_{\max}(h)$, is the length of the longest such path. The *average dilation* of h, denoted $Dil_{\text{ave}}(h)$, is the mean length of all such paths. Each of these measurements reflects the degree to which the structure of G is "stretched" by h. In this case, the closer the maximum (or average) dilation is to one, the smaller the degree of deformation of G by h.

Finally, for $e' \in E(h(G)) \subseteq E(G')$, the *congestion* of e' is the number of edges in G with images including e', i.e., the congestion of e' is $|\{e \in E(G) : e'$ is in path $h(e)\}|$. The *maximum congestion* of h, $Cong_{\max}(h)$, is the largest congestion value attained in $E(h(G))$; the *average congestion* of h, $Cong_{\text{ave}}(h)$, is the mean congestion value over $E(h(G))$. These measurements gauge the traffic flow through the edges of $h(G)$. Once again, congestion values closer to one indicate a more efficient embedding.

Although small values of each of these cost gauges are desired, we will assume here that an optimal expansion value is crucial when embedding pyramids into hypercubes. The exponential size difference between H_n and H_{n+1} is the basis for this assumption. Certain new hypercube

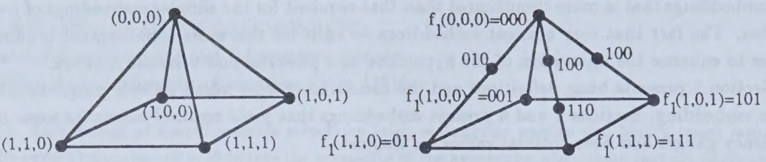


Figure 1: P_1 and its image under f_1

technologies (e.g., [15]) are not drastically affected by higher values of dilation, but the negative effects of congestion [9] have not been effectively counteracted.

Lemma 2.1 *Any embedding of P_k into a hypercube requires at least a $(2k+1)$-dimensional hypercube and, hence, an expansion of at least $(6 \cdot 4^k)/(4^{k+1} - 1) \approx 1.5$.*

Proof. $|V(P_k)| = \frac{1}{3}(4^{k+1} - 1)$ and $|V(H_{2k})| = 2^{2k} < \frac{1}{3}(4^{k+1} - 1) < 2^{2k+1} = |V(H_{2k+1})|$. $|V(H_{2k+1})|/|V(P_k)| = (6 \cdot 4^k)/(4^{k+1} - 1)$. □

Lemma 2.2 *Any embedding of P_k into a hypercube must have maximum dilation greater than one.*

Proof. Saad and Schultz [17] showed that the hypercube contains no odd cycles. The result follows, then, from the existence of triangles in P_k. □

Lemma 2.3 *For $1 \leq k \leq 3$, if $h : P_k \rightarrow H_{2k+1}$ is an embedding, then $Cong_{\max}(h) \geq 2$.*

Proof. The maximum degree of a vertex of P_k is 4, 7, and 9 for $k = 1, 2$, and 3, respectively. In each case, this value exceeds the value of $2k + 1$. The result follows. □

Although the proof above cannot be extended to values of k greater than 3, the lemma does indicate that it would be difficult, if not impossible, to develop a recursive algorithm which embeds P_k in H_{2k+1} with unit congestion.

In the next two sections, we present an algorithm that achieves congestion two with dilation three, and a second algorithm that achieves dilation two with congestion three. For lack of space, proofs in these sections will not be given, but can be obtained from the authors upon request. There appears to be a trade-off between these two gauges of embedding efficiency. The issue of cost trade-offs in graph embeddings (specifically, in tree embeddings) has been addressed by Hong, et al., in [12].

3 A CONGESTION-TWO EMBEDDING

In this section, we present a recursive algorithm that embeds the pyramid in a hypercube with optimal expansion. This algorithm has maximum congestion two and maximum dilation three.

3.1 The Algorithm

Define embedding $f_1 : P_1 \rightarrow H_3$ as illustrated in Figure 1. (For clarity, our illustrations of embedded pyramids will take the shape of their pyramid pre-images, with vertices labelled using hypercube binary representations.) Of the eight edges in P_1, one receives dilation three, two receive dilation two, and five receive dilation one. Only one edge of $f_1(P_1)$, $(000, 100)$, receives congestion two; the ten others receive unit congestion. Note that 000, the image of the apex, is adjacent to 001, an element of $V(f_1(P_1)) - f_1(V(P_1))$ and that the edge between these two vertices has congestion one under f_1. The existence of such an edge will be important in the recursive extension of this embedding to larger pyramids.

Assume now that $f_{k-1} : P_{k-1} \rightarrow H_{2k-1}$ is an embedding with $u = f_{k-1}(0,0,0)$, and that there exists $v \in V(f_{k-1}(P_{k-1})) - f_{k-1}(V(P_{k-1}))$ such that u and v are adjacent in $f_{k-1}(P_{k-1})$ and the edge (u, v) has congestion one under f_{k-1}. We show that f_{k-1} can be used to construct an embedding

$f_k : P_k \to H_{2k+1}$ such that $u' = f_k(0,0,0)$ has a neighbor $v' \in V(f_k(P_k)) - f_k(V(P_k))$ and such that edge (u', v') has congestion one.

Let $P_{k-1}(1, x_1, x_2)$ denote the subpyramid of P_k with height $k-1$ and apex $(1, x_1, x_2)$. Note that P_k has four such subpyramids. On the other hand, H_{2k+1} has four $(2k-1)$-dimensional subcubes: namely, $x_2x_1H_{2k-1}$, where $x_1, x_2 \in \{0,1\}$.

Intuitively, we will use f_{k-1} to embed each $P_{k-1}(1, x_1, x_2)$ into $x_2x_1H_{2k-1}$. The images of these embeddings will then be symmetrically arranged in a "mirrored" fashion, so that corresponding vertices on different subpyramids face each other. The (u, v) edge incident to each subpyramid's apex will then be used to construct the apex of the embedded height-k pyramid.

In order to formally describe the embedding, we introduce some notation. Define $\phi^V_{k-1} : P_{k-1} \to P_{k-1}$ to be the "vertical-reflection" embedding of P_{k-1} into itself, so that vertex $(i, x_1, x_2) \in V(P_{k-1})$ is mapped to $\phi^V_{k-1}(i, x_1, x_2) = (i, x_1, 2^i - x_2 - 1)$ and edge $(u, v) \in E(P_{k-1})$ is mapped to $\phi^V_{k-1}(u, v) = (\phi^V_{k-1}(u), \phi^V_{k-1}(v))$. Similarly, $\phi^H_{k-1} : P_{k-1} \to P_{k-1}$ is defined as the "horizontal-reflection" embedding of P_{k-1} into itself, so that $\phi^H_{k-1}(i, x_1, x_2) = (i, 2^i - x_1 - 1, x_2)$ and $\phi^H_{k-1}(u, v) = (\phi^H_{k-1}(u), \phi^H_{k-1}(v))$.

Define three additional embeddings of P_{k-1} into H_{2k-1} as follows: $f^V_{k-1} = f_{k-1} \circ \phi^V_{k-1}$, the vertical reflection of f_{k-1}; $f^H_{k-1} = f_{k-1} \circ \phi^H_{k-1}$, the horizontal reflection of f_{k-1}; and $f^{VH}_{k-1} = f_{k-1} \circ \phi^V_{k-1} \circ \phi^H_{k-1}$, the double reflection of f_{k-1}.

Define $t_{x_1,x_2} : P_{k-1}(1, x_1, x_2) \to P_{k-1}$ so that $t_{x_1,x_2}(i, x'_1, x'_2) = (i-1, x'_1 - x_1(2^{i-1}), x'_2 - x_2(2^{i-1}))$ for any vertex (i, x'_1, x'_2) in $P_{k-1}(1, x_1, x_2)$ and $t_{x_1,x_2}(u, v) = (t_{x_1,x_2}(u), t_{x_1,x_2}(v))$ for any edge (u, v) in $P_{k-1}(1, x_1, x_2)$. Thus, t_{x_1,x_2} merely transforms the coordinates of vertices in $P_{k-1}(1, x_1, x_2)$ into the corresponding coordinates in P_{k-1}.

Finally, for any two binary values b_1,b_2, define the prefix function $h_{b_2b_1} : H_{2k-1} \to H_{2k+1}$ so that $h_{b_2b_1}(x) = b_2b_1x$ and $h_{b_2b_1}(x, x') = (b_2b_1x, b_2b_1x')$ for any vertex x and edge (x, x') in H_{2k-1}.

Using this notation, we may present our recursive algorithm for the embedding $f_k : P_k \to H_{2k+1}$:

Algorithm

1. If $k = 1$, embed P_k in H_{2k+1} using f_1.
2. If $k > 1$, use f_{k-1} to define f_k as follows:

   a) Embed the subpyramid $P_1(0, 0, 0)$ in the 3-dimensional subcube of H_{2k+1} generated by $\{00u, 00v, 01u, 01v, 10u, 10v, 11u, 11v\}$ such that:

   $$\begin{array}{ll} f_k(0,0,0) = 00v, & \\ f_k(1,0,0) = 00u, & f_k(1,0,1) = 10u, \\ f_k(1,1,0) = 01u, & f_k(1,1,1) = 11u, \\ \multicolumn{2}{l}{f_k((0,0,0),(1,0,0)) = (00v, 00u),} \\ \multicolumn{2}{l}{f_k((0,0,0),(1,0,1)) = (00v, 10v, 10u),} \\ \multicolumn{2}{l}{f_k((0,0,0),(1,1,0)) = (00v, 01v, 01u),} \\ \multicolumn{2}{l}{f_k((0,0,0),(1,1,1)) = (00v, 10v, 11v, 11u).} \end{array}$$

   b) For each vertex $(1, x_1, x_2)$, embed $P_{k-1}(1, x_1, x_2)$ in $x_2x_1H_{2k-1}$ using the mapping:

   $$\begin{array}{ll} h_{00} \circ f_{k-1} \circ t_{x_1,x_2} & \text{if } x_1 = x_2 = 0 \\ h_{10} \circ f^H_{k-1} \circ t_{x_1,x_2} & \text{if } x_1 = 1, x_2 = 0 \\ h_{01} \circ f^V_{k-1} \circ t_{x_1,x_2} & \text{if } x_1 = 0, x_2 = 1 \\ h_{11} \circ f^{VH}_{k-1} \circ t_{x_1,x_2} & \text{if } x_1 = x_2 = 1 \end{array}$$

   c) For each edge $(u, v) \in E(P_k)$ with u and v in different subpyramids $P_{k-1}(1, x_1, x_2)$'s, let $f_k(u, v) = (f_k(u), f_k(v))$.

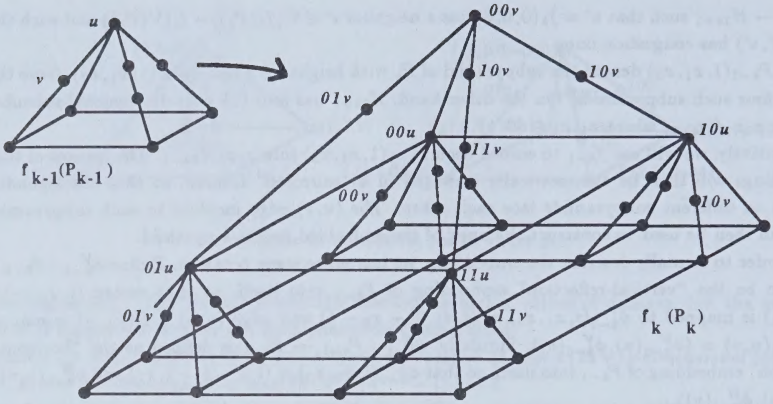


Figure 2: The recursive definition of f_k

The algorithm is illustrated in Figure 2.

3.2 Correctness

To validate the correctness of our algorithm, we must verify that the vertices are mapped in a one-to-one fashion, that the congestion-one edge (u, v) assumed in $f_{k-1}(P_{k-1})$ has a corresponding congestion-one edge (u', v') in $f_k(P_k)$, and that the edge mappings described in step 2(c) of the algorithm are well-defined. We should also note that several of the mappings described in step 2(a) are redefined in steps 2(b) and 2(c), but that these definitions are identical.

Lemma 3.1 *Using the mapping f_k, $V(P_k)$ is mapped as follows:*

$$f_k(i, x_1, x_2) = \begin{cases} 00v & \text{if } (i, x_1, x_2) = (0, 0, 0) \\ 00f_{k-1}(i-1, x_1, x_2) & \text{if } (i, x_1, x_2) \in P_{k-1}(1, 0, 0) \\ 01f_{k-1}(i-1, 2^i - x_1 - 1, x_2) & \text{if } (i, x_1, x_2) \in P_{k-1}(1, 1, 0) \\ 10f_{k-1}(i-1, x_1, 2^i - x_2 - 1) & \text{if } (i, x_1, x_2) \in P_{k-1}(1, 0, 1) \\ 11f_{k-1}(i-1, 2^i - x_1 - 1, 2^i - x_2 - 1) & \text{if } (i, x_1, x_2) \in P_{k-1}(1, 1, 1) \end{cases}$$

The following corollary illustrates the validity of step 2(c) of the algorithm.

Corollary 3.2 *If (u, v) is an edge in P_k and u and v are in different subpyramids $P_{k-1}(1, x_1, x_2)$'s, then $f_k(u)$ is adjacent to $f_k(v)$ in H_{2k+1}.*

Corollary 3.3 *For $u, v \in V(P_k)$, if $u \neq v$, then $f_k(u) \neq f_k(v)$.*

Corollary 3.3 demonstrates that f_k is an embedding of P_k into H_{2k+1}. Note that $f_k(0, 0, 0) = (00v)$ and that $01v \in V(f_k(P_k)) - f_k(V(P_k))$ such that $00v$ and $01v$ are adjacent in $f_k(P_k)$ and the edge $(00v, 01v)$ has congestion one under f_k. Our recursive algorithm is well-defined.

Some notation is needed before we can examine the dilation and congestion of f_k. The edges of P_k are composed of six disjoint sets: $E(P_{k-1}(1, 0, 0))$, $E(P_{k-1}(1, 0, 1))$, $E(P_{k-1}(1, 1, 0))$, and $E(P_{k-1}(1, 1, 1))$, as defined above; $X_k = \{(u, v) \in E(P_k) : u = (0, 0, 0)\}$, the edges joining the apex of P_k to the apex of each $P_{k-1}(1, x_1, x_2)$; and $Y_k = \{((i, 2^{i-1} - 1, x_2), (i, 2^{i-1}, x_2)) : 1 \leq i \leq k, 0 \leq x_2 \leq 2^i - 1\} \cup \{((i, x_1, 2^{i-1} - 1), (i, x_1, 2^{i-1})) : 1 \leq i \leq k, 0 \leq x_1 \leq 2^i - 1\}$, the edges joining corresponding vertices in neighboring subpyramids of the form $P_{k-1}(1, x_1, x_2)$.

3.3 Dilation

The results of the previous section enable us to determine the dilation of f_k with a minimum of effort.

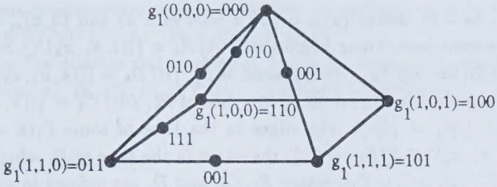


Figure 3: $g_1(P_1)$

Theorem 3.4 *Using embedding f_k, $\frac{1}{3}(4^k-1)$ edges of P_k receive dilation three, $\frac{2}{3}(4^k-1)$ edges receive dilation two, and the remaining $3\cdot 4^k - 2^{k+2}+1$ edges receive dilation one.*

Note that $Dil_{\text{ave}}(f_k) = (2^{k+2}+1)/(3\cdot 2^k)$ and that $Dil_{\text{ave}}(f_k) \to \frac{4}{3}$ as $k \to \infty$.

3.4 Congestion

Examining the amount of overlap between the images of the disjoint subsets of $E(P_k)$ and the amount of edge congestion that occurs within these sets, will allow us to specify the congestion of the embedding f_k.

Lemma 3.5

(i) If $(x_1,x_2) \neq (x_1',x_2')$, then $E(f_k(P_{k-1}(1,x_1,x_2))) \cap E(f_k(P_{k-1}(1,x_1',x_2'))) = \emptyset$.

(ii) Each edge in $f_k(Y_k)$ has congestion one.

(iii) (x_1x_2u, x_1x_2v) is the only edge that $f_k(P_{k-1}(1,x_1,x_2))$ and $E(f_k(X_k))$ have in common.

Corollary 3.6 $Cong_{\max}(f_k) = 2$.

The previous results indicate that the congestion of f_k can be determined by separating $E(P_k)$ into the six disjoint subsets mentioned in Section 3.1, observing the congestion that occurs within each subset, and taking note of the overlap that occurs between the images of the subsets.

Theorem 3.7 *Under embedding f_k, $\frac{1}{3}(2^{2k+1}-5)$ edges in $f_k(P_k)$ have congestion two, while the remaining $4^{k+1}-2^{k+2}+2$ edges have congestion one.*

Note that $Cong_{\text{ave}}(f_k) = (4^{k+2} - 3\cdot 2^{k+2} - 4)/(14\cdot 4^k - 3\cdot 2^{k+2}+1)$ and that $Cong_{\text{ave}}(f_k) \to \frac{8}{7}$ as $k \to \infty$.

4 AN OPTIMAL-DILATION EMBEDDING

The embedding algorithm of the previous section has optimal expansion, but its maximum dilation, although small, is not optimal. In this section, we present a substantially more complex recursive algorithm that embeds the pyramid into a hypercube with optimal expansion, maximum dilation two, and maximum congestion three.

4.1 The Algorithm

Define embedding $g_1 : P_1 \to H_3$ as illustrated in Figure 3. Note that five edges of P_1 receive dilation two, while the other three edges receive dilation one. Two edges of $g_1(P_1)$, (specifically, (000, 010) and (001, 101)) receive congestion two; the remaining nine edges of $g_1(P_1)$ receive unit congestion. The structure of this pyramid, including the relative positions of the dilated and congested edges, plays a significant role in our algorithm.

More generally, let $P_1(k-1,x_1,x_2)$ denote the subpyramid of P_k with height one and apex $(k-1,x_1,x_2)$. For $v \in V(H_{2k+1})$ and $1 \le p,q,r \le 2k+1$, let $H_3(v;p,q,r)$ be the 3-dimensional subcube of H_{2k+1} generated by $\{v, v^p, v^q, v^r, v^{pq}, v^{pr}, v^{qr}, v^{pqr}\}$. Using vertical, horizontal, and double reflections (as in the previous section), $P_1(k-1,x_1,x_2)$ may be embedded into $H_3(v;p,q,r)$ by any of the four embeddings g_1, g_1^H, g_1^V, and g_1^{VH}. The positions of corresponding dilated and congested edges in these embeddings exhibit the mirrored quality examined in Section 3.

For $a, b \in \{0, 1, \ldots, m-1\}$, define $\{a\}_m = \{x : x \bmod m = a\}$ and $\{a, b\}_m = \{a\}_m \cup \{b\}_m$.

The edges of P_k are composed of four disjoint sets: *(i)* $A_k = \{((i, x_1, x_2), (j, x_1', x_2')) \in E(P_k) : 0 \leq i, j \leq k-1\}$, the edges in the top P_{k-1}-subpyramid of P_k; *(ii)* $B_k = \{((k, x_1, x_2), (k-1, \lfloor \frac{x_1}{2} \rfloor, \lfloor \frac{x_2}{2} \rfloor\}$, the edges joining the top P_{k-1}-subpyramid to the base of P_k; *(iii)* $C_k = \{((k, x_1, x_2), (k, x_1', x_2')) \in E(P_k) : \lfloor \frac{x_1}{2} \rfloor = \lfloor \frac{x_1'}{2} \rfloor, \lfloor \frac{x_2}{2} \rfloor = \lfloor \frac{x_2'}{2} \rfloor\}$, the edges in the base of some $P_1(k-1, u_1, u_2)$; and *(iv)* $D_k = \{((k, x_1, x_2), (k, x_1', x_2')) \in E(P_k) - C_k\}$, the edges in the base of P_k which are not in C_k.

Note that $A_k = \cup_{i=1}^{k-1}(B_i \cup C_i \cup D_i)$, where B_i, C_i, and D_i are defined in an analogous fashion.

For each vertex v in the base of P_k, let $\alpha^i(v)$ denote the ancestor of v on level i of P_k (i.e., $\alpha^i(k, x_1, x_2) = (i, \lfloor x_1/2^{k-i} \rfloor, \lfloor x_2/2^{k-i} \rfloor)$).

Using this notation, we may present our recursive algorithm for the embedding $g_k : P_k \rightarrow H_{2k+1}$:

Algorithm

1. If $k = 1$, embed P_k in H_{2k+1} using g_1.
2. If $k > 1$, use g_{k-1} to define g_k as follows:
    a) Embed the top P_{k-1}-subpyramid of P_k in H_{2k+1} using $h \circ g_{k-1}$, where $h : H_{2k-1} \rightarrow H_{2k+1}$ such that $h(x) = 00x$.

    b) For each vertex $u = (k-1, x_1, x_2)$, embed $P_1(u)$ in $H_3(00g_{k-1}(u); 2k-1, 2k, 2k+1)$ using the mapping:

    g_1 if $x_1, x_2 \in \{0\}_2$
    g_1^V if $x_1 \in \{0\}_2, x_2 \in \{1\}_2$
    g_1^H if $x_1 \in \{1\}_2, x_2 \in \{0\}_2$
    g_1^{VH} if $x_1, x_2 \in \{1\}_2$

    c) For any edge $e = (u, v) \in D_k$, let $e' = (u', v') = (\alpha^{k-1}(u), \alpha^{k-1}(v))$.
    If $g_k(e') = (g_k(u'), g_k(u')^i, g_k(v'))$ for some $i \in \{1, \ldots, 2k+1\}$, then embed e in H_{2k+1} using $g_k(e) = (g_k(u), g_k(u)^i, g_k(v))$.
    Otherwise, if $g_k(e') = (g_k(u'), g_k(v'))$, then let $g_k(e) = (g_k(u), g_k(v))$.

Note that this algorithm uses a top-down approach to recursively construct the embedding, unlike the bottom-up approach used in the previous algorithm. Embedding f_k used four mirrored copies of $f_{k-1}(P_{k-1})$ and constructed a new apex for a larger pyramid; g_k constructs an array of height-one pyramids from the base of $g_{k-1}(P_{k-1})$, forming a new base for a larger pyramid.

4.2 Correctness

As with our previous algorithm, we must validate the correctness of this algorithm by verifying that the vertex mapping is one-to-one and that the edge mappings in step 2(c) of the algorithm are well-defined.

Lemma 4.1 *Under the mapping g_k, $V(P_k)$ is mapped as follows:*

$$g_k(i, x_1, x_2) = 00g_{k-1}(i, x_1, x_2) \qquad \text{if } 0 \leq i \leq k-1$$

$$g_k(k, x_1, x_2) = \begin{cases} 11g_{k-1}(u) & \text{if } x_1, x_2 \in \{0, 3\}_4 \\ 10g_{k-1}(u) & \text{if } x_1 \in \{0, 3\}_4, x_2 \in \{1, 2\}_4 \\ 01(g_{k-1}(u))^{2k-1} & \text{if } x_1 \in \{1, 2\}_4, x_2 \in \{0, 3\}_4 \\ 10(g_{k-1}(u))^{2k-1} & \text{if } x_1, x_2 \in \{1, 2\}_4 \end{cases}$$

where $u = \alpha^{k-1}(k, x_1, x_2) = (k-1, \lfloor \frac{x_1}{2} \rfloor, \lfloor \frac{x_2}{2} \rfloor)$.

For $u, v \in V(P_k)$, let $S(u, v)$ denote the set $\{i : 1 \leq i \leq 2k+1, g_k(u)$ differs from $g_k(v)$ in bit $i\}$.

The following lemma illustrates the validity of step 2(c) of the algorithm.

Lemma 4.2 *If $e = (u, v) \in D_k$ and $e' = (u', v') = (\alpha^{k-1}(u), \alpha^{k-1}(v))$, then $S(u, v) = S(u', v')$ and $|S(u, v)| \leq 2$.*

The following lemma demonstrates that g_k is, in fact, an embedding of P_k into H_{2k+1}.

Lemma 4.3 *For $u, v \in V(P_k)$, if $u \neq v$, then $g_k(u) \neq g_k(v)$.*

In order to examine the dilation and congestion under g_k, we need to analyze the embedding's effect on the four subsets $A_k, B_k, C_k, D_k \subseteq E(P_k)$, as defined in Section 4.1.

Lemma 4.4

(i) If $e \in A_k$, then $g_k(e) = 00g_{k-1}(e)$.

(ii) If $e = (u, v) = ((k-1, \lfloor \frac{x_1}{2} \rfloor, \lfloor \frac{x_2}{2} \rfloor), (k, x_1, x_2)) \in B_k$, then

$$g_k(e) = \begin{cases} (00g_{k-1}(u), 01g_{k-1}(u), 11g_{k-1}(u)) & \text{if } x_1, x_2 \in \{0,3\}_4 \\ (00g_{k-1}(u), 10g_{k-1}(u)) & \text{if } x_1 \in \{0,3\}_4, x_2 \in \{1,2\}_4 \\ (00g_{k-1}(u), 01g_{k-1}(u), 01(g_{k-1}(u))^{2k-1}) & \text{if } x_1 \in \{1,2\}_4, x_2 \in \{0,3\}_4 \\ (00g_{k-1}(u), 00(g_{k-1}(u))^{2k-1}, 10(g_{k-1}(u))^{2k-1}) & \text{if } x_1, x_2 \in \{1,2\}_4 \end{cases}$$

(iii) If $e = (u, v) = ((k, x_1, x_2), (k, x_1 + 1, x_2)) \in C_k$, then

$$g_k(e) = \begin{cases} (g_k(u), (g_k(u))^{2k-1}) & \text{if } x_1 \in \{0\}_2, x_2 \in \{1,2\}_4 \\ (g_k(u), (g_k(u))^{2k-1}, (g_k(u))^{2k-1,2k+1}) & \text{if } x_1 \in \{0\}_4, x_2 \in \{0,3\}_4 \\ (g_k(u), (g_k(u))^{2k+1}, (g_k(u))^{2k+1,2k-1}) & \text{if } x_1 \in \{2\}_4, x_2 \in \{0,3\}_4 \end{cases}$$

(iv) If $e = (u, v) = ((k, x_1, x_2), (k, x_1, x_2 + 1)) \in C_k$, then

$$g_k(e) = \begin{cases} (g_k(u), (g_k(u))^{2k}) & \text{if } x_1 \in \{0,3\}_4, x_2 \in \{0\}_2 \\ (g_k(u), (g_k(u))^{2k}, (g_k(u))^{2k,2k+1}) & \text{if } x_1 \in \{1,2\}_4, x_2 \in \{0\}_4 \\ (g_k(u), (g_k(u))^{2k+1}, (g_k(u))^{2k+1,2k}) & \text{if } x_1 \in \{1,2\}_4, x_2 \in \{2\}_4 \end{cases}$$

Corollary 4.5 *For any x_1, x_2, x_1', x_2' with $0 \leq x_1, x_2, x_1', x_2' \leq 2^{k-1} - 1$, with $(x_1, x_2) \neq (x_1', x_2')$, $g_k(P_1(k-1, x_1, x_2))$ and $g_k(P_1(k-1, x_1', x_2'))$ are vertex-disjoint and edge-disjoint.*

Complications arise when attempts are made to analyze the image under g_k of an edge $e = (u, v)$ in set D_k. Such an edge joins the base vertices of neighboring pyramids of the form $P_1(k-1, x_1, x_2)$. Lemma 4.2 indicates that the sequence of bit differences in path $g_k(e)$ will reflect the bit differences in the corresponding path between the g_k-images of the ancestors of u and v. Determining these bit differences will enable us to complete the specification of $g_k(E(P_k))$ and analyze the dilation and congestion of g_k.

For any edge $(u, v) \in D_k$, define $\sigma(u, v)$ to be the minimum value of i $(0 \leq i \leq k-1)$ such that $\alpha^i(u) \neq \alpha^i(v)$ (i.e., $\sigma(u, v)$ is the level at which the ancestors of u and v have the same parent).

Lemma 4.6

(i) If $e = (u, v) = ((k, x_1, x_2), (k, x_1 + 1, x_2)) \in D_k$, then, letting $u'' = (\sigma(u, v), x_1'', x_2'')$ be the ancestor of u at level $\sigma(u, v)$ and $v'' = (\sigma(u, v), x_1'' + 1, x_2'')$ be the ancestor of v at level $\sigma(u, v)$, we have:

$$g_k(e) = \begin{cases} (g_k(u), (g_k(u))^{2\sigma(u,v)-1}) & \text{if } x_1'' \in \{0\}_2, x_2'' \in \{1,2\}_4 \\ (g_k(u), (g_k(u))^{2\sigma(u,v)-1}, (g_k(u))^{2\sigma(u,v)-1,2\sigma(u,v)+1}) & \text{if } x_1'' \in \{0\}_4, x_2'' \in \{0,3\}_4 \\ (g_k(u), (g_k(u))^{2\sigma(u,v)+1}, (g_k(u))^{2\sigma(u,v)+1,2\sigma(u,v)-1}) & \text{if } x_1'' \in \{2\}_4, x_2'' \in \{0,3\}_4 \end{cases}$$

(ii) If $e = (u, v) = ((k, x_1, x_2), (k, x_1, x_2 + 1)) \in D_k$, then, letting $u'' = (\sigma(u, v), x_1'', x_2'')$ be the ancestor of u at level $\sigma(u, v)$ and $v'' = (\sigma(u, v), x_1'', x_2'' + 1)$ be the ancestor of v at level $\sigma(u, v)$, we have:

$$g_k(e) = \begin{cases} (g_k(u), (g_k(u))^{2\sigma(u,v)}) & \text{if } x_1'' \in \{0,3\}_4, x_2'' \in \{0\}_2 \\ (g_k(u), (g_k(u))^{2\sigma(u,v)}, (g_k(u))^{2\sigma(u,v),2\sigma(u,v)+1}) & \text{if } x_1'' \in \{1,2\}_4, x_2'' \in \{0\}_4 \\ (g_k(u), (g_k(u))^{2\sigma(u,v)+1}, (g_k(u))^{2\sigma(u,v)+1,2\sigma(u,v)}) & \text{if } x_1'' \in \{1,2\}_4, x_2'' \in \{2\}_4 \end{cases}$$

4.3 Dilation

The disjoint subsets $A_k, B_k, C_k, D_k \subseteq E(P_k)$ defined in Section 4.1 may now be utilized to determine the dilation of g_k.

Theorem 4.7 *Under embedding* g_k, $\frac{7}{3}4^k - 2^{k+1} - \frac{1}{3}$ *edges receive dilation two, and the remaining* $\frac{5}{3}4^k - 2^{k+1} + \frac{1}{3}$ *edges receive dilation one.*

Note that $Dil_{\text{ave}}(g_k) = (19 \cdot 2^k + 1)/(3 \cdot 2^{k+2})$ and that $Dil_{\text{ave}}(g_k) \to \frac{19}{12}$ as $k \to \infty$.

4.4 Congestion

To analyze the congestion of our new embedding, we first examine the overlap between the images of the disjoint subsets $A_k, B_k, C_k, D_k \subseteq E(P_k)$; any edge in the hypercube that is in the image of more than one of these subsets will be shown to have a correspondingly higher congestion. Determining the degree of edge congestion that occurs within these subsets will then enable us to specify the congestion of g_k.

For $E' \subseteq E(P_k)$, let $g_k(E')$ denote the set of edges $\{e \in E(g_k(P_k)) : e \text{ is on path } g_k(e') \text{ for some } e' \in E'\}$.

Lemma 4.8 *For* $1 \le i \le k$,

(i) $g_k(A_i) \cap g_k(B_i) = \{(g_k(v), (g_k(v))^{2i-1}) : v = (i-1, x_1, x_2); 0 \le x_1, x_2 \le 2^{i-1} - 1\}$.
(ii) $g_k(A_i) \cap g_k(C_i) = g_k(A_i) \cap g_k(D_i) = \emptyset$.
(iii) $g_k(B_i) \cap g_k(C_i) = \{(g_k(v), (g_k(v))^{2i+1}) : v = (i, x_1, x_2); x_1, x_2 \in \{1,2\}_4\}$.
(iv) $g_k(B_i) \cap g_k(D_i) = \{(g_k(v), (g_k(v))^{2i-1}) : v = (i, x_1, x_2); x_1 \in \{2,5\}_8; x_2 \in \{0,7\}_8\}$.
(v) $g_k(C_i) \cap g_k(D_i) = \{(g_k(v), (g_k(v))^{2i-1}) : v = (i, x_1, x_2); x_1 \in \{2,5\}_8; x_2 \in \{1,2\}_4\}$.

Corollary 4.9 *For* $1 \le i, j \le k$,

(i) if $|i-j| > 1$, *then* $g_k(B_i) \cap g_k(B_j) = \emptyset$
(ii) if $i \ne j$, *then* $g_k(C_i) \cap g_k(C_j) = \emptyset$, $g_k(C_i) \cap g_k(D_j) = \emptyset$, *and* $g_k(D_i) \cap g_k(D_j) = \emptyset$.
(iii) $g_k(B_i) \cap g_k(C_j) \cap g_k(D_j) = \emptyset$.

Lemma 4.8 indicates how much congestion takes place between the images of the disjoint sets of pyramid edges. To complete an analysis of the congestion of g_k, we must examine the amount of congestion that takes place within these individual sets.

We say that $e \in g_k(E')$ has congestion t *within* $g_k(E')$ if $|\{e' \in E' : e \text{ is on path } g_k(e')\}| = t$. Finally, we define $\tau_t^k(E') = \{e \in g_k(E') : e \text{ has congestion } t \text{ within } g_k(E')\}$.

Lemma 4.10 *Using the definition of* $\tau_t^k(E')$ *given above, we have, for* $1 \le i \le k$,

(i) $\tau_2^k(B_i) = \{(g_k(v), (g_k(v))^{2i}) : v = (i-1, x_1, x_2); 0 \le x_1, x_2 \le 2^{i-1} - 1\}$;
$\tau_1^k(B_i) = g_k(B_i) - \tau_2^k(B_i)$.
(ii) $\tau_1^k(C_i) = g_k(C_i)$.
(iii) $\tau_2^k(D_i) = \{(g_k(v), (g_k(v))^{2i-1}) : v = (i, x_1, x_2); x_1, x_2 \in \{2,5\}_8\}$;
$\tau_1^k(D_i) = g_k(D_i) - \tau_2^k(D_i)$.

The previous results enable us to completely specify the congestion of every element of $g_k(P_k)$.

Theorem 4.11 *Under embedding* g_k, $\frac{2}{3}(4^{k-1} - 1)$ *of the edges in* $g_k(P_k)$ *have congestion three,* $\frac{1}{3}(11 \cdot 4^{k-1} - 5)$ *edges have congestion two, and the remaining* $4^{k+1} - 3 \cdot 2^{k+1} + 5$ *edges have congestion one.*

Note that $Cong_{\text{ave}}(g_k) = (19 \cdot 4^k - 9 \cdot 2^{k+1} - 1)/(61 \cdot 4^{k-1} - 9 \cdot 2^{k+1} + 8)$ and that $Cong_{\text{ave}}(g_k) \to \frac{76}{61} \approx 1.25$ as $k \to \infty$.

5 CONCLUDING REMARKS

The two algorithms presented in this paper provide efficient embeddings of the pyramid into the hypercube. Each embedding has optimal expansion cost, thus minimizing the number of hypercube processors which are unused or used only for data communication. The g_k embedding supports optimal maximum dilation, while the f_k embedding has maximum dilation that is near-optimal, but average dilation that is noticeably smaller than that in g_k. Consequently, both algorithms display a favorable tendency to avoid significantly modifying the pyramid structure. Neither embedding suffers problems with traffic flow; we conjecture that f_k has optimal maximum congestion (under our mandatory optimal expansion constraint), while g_k supports near-optimal congestion.

Several questions merit further research in this area. For example, our results do not eliminate the possibility of optimizing all three cost gauges when embedding pyramids into hypercubes, but they do suggest that efficiency trade-offs may exist, as indicated in our conjecture. Also, it is possible that our restriction to embeddings that optimize expansion could be relaxed if attempts are made to embed a number of disjoint pyramids into a single large hypercube.

REFERENCES

[1] B. Becker and H.U. Simon, "How Robust is the n-Cube?," *Proc. 27th Annu. IEEE Symp. Foundations Comp. Sci.*, October 1986, pp. 283–291.

[2] F. Berman and L. Snyder, "On Mapping Parallel Algorithms into Parallel Architectures," *Journal of Parallel and Distributed Computing,* Vol. 4, No. 5, October 1987, pp. 439–458.

[3] S.N. Bhatt and I.C.F. Ipsen, "How to Embed Trees in Hypercubes," Yale University Research Report RR-443, December 1985.

[4] S. Bhatt, F. Chung, T. Leighton, and A. Rosenberg, "Optimal Simulations of Tree Machines," *Proc. 27th Annu. Symp. Foundations Comp. Sci.*, October 1986, pp. 274–282.

[5] T.F. Chan and Y. Saad, "Multigrid Algorithms on the Hypercube Multiprocessor," *IEEE Trans. on Comp.*, Vol. c-35, No. 11, November 1986, pp. 969–977.

[6] J.H. Chang, O.H. Ibarra, T.C. Pong, and S.M. Sohn, "Two-Dimensional Convolution on a Pyramid Computer," *Proc. of the 1987 Internat. Conf. on Parallel Processing,* August 1987, pp. 780–782.

[7] G.Cybenko, D.W. Krumme, and K.N. Venkataraman, "Fixed Hypercube Embedding," *Information Processing Letters,* Vol. 25, April 1987, pp. 35–39.

[8] S.R. Deshpande and R.M. Jenevein, "Scalability of a Binary Tree on a Hypercube," *Proc. of the 1986 Internat. Conf. on Parallel Processing,* August 1986, pp. 661–668.

[9] J. Hastad, T. Leighton, and M. Newman, "Reconfiguring a Hypercube in the Presence of Faults," *Proc. 19th Annu. ACM Symp. Theory of Computing,* May 1987, pp. 274–284.

[10] C.T. Ho and S.L. Johnsson, "Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes," *Proc. of the 1986 Internat. Conf. on Parallel Processing,* August 1986, pp. 640–648.

[11] C.T. Ho and S.L. Johnsson, "On the Embedding of Arbitrary Meshes in Boolean Cubes with Expansion Two Dilation Two," *Proc. of the 1987 Internat. Conf. on Parallel Processing,* August 1987, pp. 188–191.

[12] J.W. Hong, K. Mehlhorn, and A. Rosenberg, "Cost Trade-offs in Graph Embeddings, with Applications," *J. ACM,* Vol. 30, No. 4, October 1983, pp. 709–728.

[13] Y.E. Ma and L. Tao, "Embeddings among Toruses and Meshes," *Proc. of the 1987 Internat. Conf. on Parallel Processing,* August 1987, pp. 178–187.

[14] R. Miller and Q.F. Stout, "Data Movement Techniques for the Pyramid Computer," *SIAM J. Computing,* Vol. 16, No. 1, February 1987, pp. 38–60.

[15] S.F. Nugent, "The iPSC/2 Direct-Connect Communications Technology," *Proc. Third Conf. on Hypercube Computers and Applications,* January 1988.

[16] Y. Saad and M.H. Schultz, "Data Communication in Hypercubes," Yale University Research Report RR-428, October 1985.

[17] Y. Saad and M.H. Schultz, "Topological Properties of Hypercubes," Yale University Research Report RR-3389, June 1985.

[18] C.L. Seitz, "The Cosmic Cube," *C. ACM,* Vol. 28, No. 1, January 1985, pp. 22–33.

[19] Q.F. Stout. "Hypercubes and Pyramids," V. Cantoni and S. Levialdi eds., *Pyramidal Systems for Computer Vision,* Springer-Verlag, 1986, pp. 75–89.

[20] Q.F. Stout, "Sorting, Merging, Selecting, and Filtering on Tree and Pyramid Machines," *Proc. 1983 Internat. Conf. on Parallel Processing,* 1983, pp. 214–221.

[21] A.Y. Wu, "Embedding of Tree Networks into Hypercubes," *Journal of Parallel and Distributed Computing,* Vol. 2, No. 3, August 1985, pp. 238–249.

Optimization of Computation Time for Systolic Arrays

Yiwan Wong† and Jean-Marc Delosme††
†Department of Computer Science
††Department of Electrical Engineering
Yale University
New Haven, CT 06520

ABSTRACT

The time performance of a systolic array implementation of an algorithm is given by the product of the number of systolic cycles required to perform the computation and the cycle time. In this paper, the determination of a linear schedule which minimizes this product is formulated as a combinatorial optimization problem. An efficient branch-and-bound method is proposed for solving the problem.

categories: combinatorial optimization, parallel and distributed computation

1. INTRODUCTION

Advances in fabrication technology and design methodology enable the design of highly parallel, special purpose architectures for handling computation intensive problems. Among those architectures, the *systolic arrays* introduced by Kung and Leiserson [1] have received much attention due to their relative simplicity and yet wide applicability. A systolic array is a collection of processors, possibly of different functionalities, locally connected in a regular pattern. It is particularly suited for the implementation of algorithms which demonstrate regular data dependence (or equivalently, data flow) patterns; for example, many of the signal/image processing and matrix computation algorithms [2].

The time performance of a systolic implementation of an algorithm is equal to the product of two related quantities: the number of *systolic cycles* required to execute the algorithm, n_{sys}, and the maximum duration of a systolic cycle, the cycle time t_{sys}. The value of n_{sys} is a function of the size of the problem being solved and the data dependence structure among the computations. The value of t_{sys}, on the other hand, depends on the complexity of the computations performed within a systolic cycle, the precedence among these computations, and the amount of hardware resources available on the processors. There is a tradeoff between these two quantities as can be demonstrated by the following example.

> Consider the one-dimensional convolution algorithm which computes the output signal y_i, $i = 1 \cdots N$, from the input signal x_j, $j = 0 \cdots N-1$, and the set of weights, w_j, $j = 1 \cdots k$, according to the equation
>
> $$y_i = \sum_{j=1}^{k} w_j \times x_{i-j} \ .$$
>
> Assume that each processor has a multiplier and an adder which can operate in parallel, and some registers for storage. Furthermore, let $T_\times$ and T_+ be the propagation delay of the multiplier and the adder respectively. Two systolic arrays that implement this algorithm are shown in Fig. 1. Array (I) uses one less systolic cycle than array (II) but its cycle time is larger than that of (II) by T_+. Hence, if time performance is the only cost criterion, array (II) should be selected.

Previous works in systolic array synthesis (see [3] for a survey) do not take such tradeoff into account since only the value of n_{sys} is used as the measurement of time performance for the arrays derived. Such a model is valid when each processor in the array is assumed to have a single functional unit; the cycle time is then a constant equal to the total time needed to perform *sequentially* the computations assigned to a processor within one systolic cycle. However, as can be noted from the above example, even for the simple case of two functional units per processor the model is far from being satisfactory.

In [4] it is shown that the feasible *linear schedules* for a set of computations with regular (*uniform*) dependences can be determined from solving an integer linear programming (ILP) problem. Once a linear schedule has been selected, the minimum value of t_{sys} (with respect to the selected schedule) can be obtained through a register reallocation procedure called *retiming* [5]. To find the linear schedule that minimizes the product $n_{sys} \times t_{sys}$ is therefore a combinatorial optimization problem. In this paper, we present an efficient branch-and-bound method for solving this problem.

To avoid the task of resource allocation, which is *NP–hard* [6], we assume that the processors on the array contain an "adequate" number of functional units to achieve the required level of parallelism. This assumption is valid for very large scale integration (VLSI) and wafer scale integration (WSI) implementations of systolic arrays. In the concluding section, we will discuss the difficulties involved in solving the same optimization problem when the number of functional units per processor is fixed.

The rest of the paper is organized as follows. In the next section, we provide the background essential for the discussion. Section 3 formulates the optimization problem and describes the branch-and-bound method in detail. Preliminary results on the performance of the branch-and-bound method are reported in Section 4.

2. PRELIMINARIES

To facilitate the presentation, we will use the one-dimensional convolution algorithm stated in the introduction as an example throughout this section. For brevity, in-depth discussions of the topics are omitted here. Interested readers should consult the cited references for more details.

Some basic definitions in graph theory are necessary for the rest of the discussion: A *path* from $\mathbf{u}_1$ to $\mathbf{u}_p$ in a directed graph is a non-null sequence of directed edges $\mathbf{u}_1\mathbf{u}_2$, $\mathbf{u}_2\mathbf{u}_3$, $\cdots$, $\mathbf{u}_{p-1}\mathbf{u}_p$. A *circuit* is a closed path where the end points coincide. A circuit is *elementary* if no two of its nodes, except for the end points, coincide. An edge is called a *zero-edge* if its weight is zero (or null, if the weight is a vector quantity). A *zero-path* is a path which contains only zero-edges, and a *zero-circuit* is a circuit with the same property.

Uniform recurrence equations

Algorithms which possess uniform (or *shift-invariant*) data dependences can be expressed in the form of *uniform recurrence equations* (URE's) [4]. The one-dimensional convolution algorithm can be rewritten as a set of four URE's for computing the set of *variables* $\{\mathbf{y},\, \mathbf{p},\, \mathbf{w},\, \mathbf{x}\}$,

$$\mathbf{y}_{ij} = \mathbf{y}_{i,j-1} + \mathbf{p}_{ij} \qquad\qquad \mathbf{p}_{ij} = \mathbf{w}_{i-1,j} \times \mathbf{x}_{i-1,j-1}$$
$$\mathbf{w}_{ij} = \begin{cases} \mathbf{w}_{i-1,j} & \text{if } i-1 \neq 0 \\ w_j & \text{otherwise} \end{cases} \qquad \mathbf{x}_{ij} = \begin{cases} \mathbf{x}_{i-1,j-1} & \text{if } i-1 \neq 0 \text{ and } j-1 \neq 0 \\ x_{i-j} & \text{otherwise} \end{cases} \tag{1}$$

where $\mathbf{y}_{i0} \equiv 0$, and $\{\mathbf{y}_{ik} \equiv y_i \,,\ i = 1 \cdots N\}$ is the set of outputs.

The collection of index points, $\{\ (i\,,\ j)\ \}$, upon which the set of URE's is defined is called the *range* of the algorithm. In the above example, the range is a rectangle of size $N \times k$ with its lower left hand corner located at index point (1, 1). In general, the range is assumed to be an n-dimensional polyhedron with integral vertices (an integral polyhedron). Since the same set of recurrence equations is evaluated at each index point within the range, we shall treat an index point as a unit of computation of the set of URE's.

The data dependences among the computations defined by a set of URE's can be depicted by a directed graph (see Fig. 2a) whose structure can be captured by a compact *dependence graph* (DG) in which there is a node for each of the variables defined in the set of URE's and the edges correspond to the dependences among the variables [4] (Fig. 2b). The weight on any of the directed edges in the DG is equal to the vector difference, called the *dependence vector*, between the indices of the two variables the edge connects. The notation $d_{\mathbf{uv}}$ is used to denote the dependence vector from variable $\mathbf{u}$ to variable $\mathbf{v}$. There may be multiple edges (with different edge weights) between a pair of nodes.

For example, the data dependence between the variables $\mathbf{p}$ and $\mathbf{x}$ in (1) is given by the dependence vector $d_{\mathbf{px}} = (1\ \ 1)$ which specifies that the computation of variable $\mathbf{p}$ at any index point R in the range uses the value of variable $\mathbf{x}$ computed at the index point $(R\, -\, d_{\mathbf{px}})$. The notation $\mathbf{p}_R \rightarrow \mathbf{x}_{(R\,-\,d_{\mathbf{px}})}$ is used to denote such data dependence among the index points in the range.

Reindexing

The dependence vectors in a set of URE's can be modified, while preserving the functionality of the algorithm represented by the set of URE's, by a process called reindexing [7]. Let $\{\delta_{\mathbf{u}}\}$ be a set of integral n-vectors, one for each variable $\mathbf{u}$ in the URE's. Reindexing is achieved by offsetting, for each $\mathbf{u}$, the set of index points upon which the variable $\mathbf{u}$ is computed by $\delta_{\mathbf{u}}$. Hence, an instance of a variable $\mathbf{u}$ which is originally computed at an index point, R say, will be computed at index point $(R\, -\, \delta_{\mathbf{u}})$ after reindexing. If variable $\mathbf{u}$ depends on variable $\mathbf{v}$ with dependence vector $d_{\mathbf{uv}}$ then, after reindexing,

$$\mathbf{u}_{(R-\delta_{\mathbf{u}})} \rightarrow \mathbf{v}_{(R-d_{\mathbf{uv}}-\delta_{\mathbf{v}})} \, .$$

Thus, through reindexing, the dependence vector $d_{\mathbf{uv}}$ becomes $\overline{d}_{\mathbf{uv}}$ where

$$\overline{d}_{\mathbf{uv}} = d_{\mathbf{uv}} + \delta_{\mathbf{v}} - \delta_{\mathbf{u}} \, .$$

For example, by picking $\delta_{\mathbf{y}} = (0 \;\; 0)$, $\delta_{\mathbf{p}} = \delta_{\mathbf{w}} = \delta_{\mathbf{x}} = (1 \;\; 0)$, the DG of the one-dimensional convolution algorithm becomes the one shown in Fig. 3a. Note that the sum of the dependence vectors in any elementary circuit in the DG is independent of reindexing.

Reindexing also modifies the shape and the size of the computation range; the overall range is the *union* of the shifted computation ranges of the variables. By assuming that the magnitudes of the offset vectors $\{\delta_{\mathbf{u}}\}$ are "small" compared to the size of the original range, the change in size can often be considered as insignificant. For our example, reindexing changes the size of the range from $N \times k$ to $(N+1) \times k$.

Linear Scheduling of URE's

Given a set of URE's (and a chosen reindexing) with dependence vectors $\{\overline{d}_{\mathbf{uv}}\}$, a *feasible* schedule is an ordering of the index points within the computation range such that the precedence relationships induced by the dependence vectors are observed. Due to the regular dependence structure of URE's, linear schedules are particularly easy to derive and simple to implement. More importantly, linear schedules (and linear space assignment functions) preserve the shift invariance property of the data dependences and, hence, the resultant parallel implementations require only regular interconnections and homogeneous control — two of the key features of systolic arrays.

A linear schedule is specified by an integral column n-vector τ whereby the computations at an index point R in the range are scheduled for execution at cycle $R\tau$. Therefore, to observe the data dependences, a feasible linear schedule τ has to satisfy the constraints

$$R\tau \geq (R - \overline{d}_{\mathbf{uv}})\tau \qquad \forall\, R \in \text{range of computation} \, ,$$

if variable $\mathbf{u}$ depends on variable $\mathbf{v}$ with dependence vector $\overline{d}_{\mathbf{uv}}$. In other words, τ is a feasible linear schedule of the set of URE's if

$$\overline{d}_{\mathbf{uv}}\tau \geq 0 \qquad \forall\, \overline{d}_{\mathbf{uv}} \, . \tag{3}$$

The value of the inner product $\overline{d}_{\mathbf{uv}}\tau$ can be interpreted as the number of systolic cycles elapsed from the cycle on which the value of $\mathbf{v}$ at index point $(R - \overline{d}_{\mathbf{uv}})$ is computed to the cycle on which this value is used for computing $\mathbf{u}$ at index point R. In particular, a zero value for the inner product implies that the computations of $\mathbf{v}$ at $(R - \overline{d}_{\mathbf{uv}})$ and $\mathbf{u}$ at R have to occur *sequentially* within the *same* systolic cycle (but not necessarily on the same processor) and hence, the cycle time is increased.

The relationship between a linear schedule τ and the resultant cycle time can be observed from the *register graph* (RG), which may be derived from the DG and τ with the following procedure: (1) associate to each node of the DG an integer value equal to the propagation (or computation) delay of the functional unit that computes the variable the node represents, and (2) replace the weight of each edge in the DG, $\overline{d}_{\mathbf{uv}}$ say, with $\overline{d}_{\mathbf{uv}}\tau$. Each node on the RG represents a functional unit and the weights on the edges of the RG are the number of registers to be inserted between connected functional units for synchronization. For ease of discussion, we will call these weights "registers" from now on. For our example, let $\tau = (1 \;\; 1)^{\mathrm{T}}$; the RG's of the DG's in Fig. 2b (before reindexing) and Fig. 3a (after reindexing) are shown in Fig. 3b and 3c, respectively.

Let T_{pdmax} be the maximum propagation delay among the functional units in the RG. The cycle time associated with a linear schedule τ is

$$t_{sys} = MAX\left[\, T_{pdmax} \, , \; \underset{\text{every zero-path in RG}}{MAX} \; (\text{sum of propagation delays on the zero-path}) \,\right] . \tag{4}$$

If τ is chosen such that the RG contains a zero-circuit, the corresponding schedule exhibits *computation rippling*: a sequence of computations, whose length depends on the size of the computation range, is scheduled to be computed sequentially (due to data dependences) within the same systolic cycle. Hence, the cycle time of the resultant schedule is proportional to the size of the range, which can be arbitrarily large. For example, if we had picked $\tau = (1 \;\; 0)^{\mathrm{T}}$ for the DG shown in Fig. 2b, the resultant RG would contain a zero-circuit (which is a self-loop) around the node which denotes the addition operation for computing variable $\mathbf{y}$ using the values of $\mathbf{y}$ and $\mathbf{p}$. For any index point (i, j) within the computation range, the data dependence of variable $\mathbf{y}$ on other instances of $\mathbf{y}$ is

$$\mathbf{y}_{ij} \rightarrow \mathbf{y}_{i,j-1} \rightarrow \mathbf{y}_{i,j-2} \rightarrow \cdots \rightarrow \mathbf{y}_{i0},$$

but all these $\mathbf{y}$ values are scheduled by τ for computation at cycle i. Thus, the cycle time associated with the given τ must be proportional to k (= maximum value of j), the size parameter of the computation range. The definition of t_{sys} in (4) is consistent with this observation since a zero-circuit in the RG results in $t_{sys} = \infty$.

In order to avoid computation rippling, a feasible linear schedule $\tau \in \mathbf{Z}^n$ has to satisfy the additional constraint

$$S_j \tau = l_j \geq 1, \; j = 1 \cdots m \tag{5}$$

where S_j is the sum of the dependence vectors (or the edge weights) on the jth elementary circuit in the DG, l_j is an integer, and m is the number of elementary circuits. The inner product $S_j \tau$ can also be interpreted as the total number of registers on the (corresponding) jth elementary circuit in the RG. As mentioned earlier, the value of S_j is independent of reindexing.

The value of n_{sys}, which is the number of cycles between the first and last cycle at which computation occurs, depends on the choice of τ and the size of the computation range,

$$n_{sys} = 1 + \underset{R_1, R_2 \in \text{range}}{MAX} (R_1 - R_2)\tau .$$

Since the range is an integral polyhedron, the above expression can be simplified to

$$n_{sys} = 1 + \underset{v_1, v_2 \in V}{MAX} (v_1 - v_2)\tau ,$$

where V is the set of vertices of the computation range.

The value of l_j in (5) constrains the choice of τ such that there are *exactly* l_j registers in the jth elementary circuit in the resultant RG. The choice of l_j affects the values of t_{sys} and n_{sys} indirectly. A large l_j could decrease the value of t_{sys} because there is less chance for a long zero-path to exist in the RG. On the other hand, a small l_j could lead to a τ of smaller magnitude and may, therefore, reduce the value of n_{sys}.

The foregoing discussion seems to imply that a reindexing has to be chosen a priori in order to determine a feasible linear schedule (refer to inequality (3)). Fortunately, this is not the case. In [7], it is shown that (5) alone is a necessary and sufficient condition for τ to be a feasible linear schedule; for each τ which satisfies (5), there is a reindexing such that (3) holds, and vice versa. Thus, for a given set of values $\{l_j\}$, which is yet to be determined, the optimal τ which minimizes n_{sys} is obtained from solving the ILP:

$$\left\{ \begin{array}{ll} \text{minimize} & 1 + \underset{v_1, v_2 \in V}{MAX} (v_1 - v_2)\tau \\ \text{subject to} & S_j \tau = l_j , \; j = 1 \cdots m \\ & \tau \in \mathbf{Z}^n \end{array} \right. \equiv \left\{ \begin{array}{ll} \text{minimize} & 1 + Y \\ \text{subject to} & Y - (v_i - v_j)\tau \geq 0, \quad \forall \, v_i , \, v_j \in V \\ & S_j \tau = l_j , \; j = 1 \cdots m \\ & \tau \in \mathbf{Z}^n \end{array} \right.$$

Retiming

Given a register graph RG, the cycle time t_{sys} can be minimized by rearranging the registers on the graph. This is accomplished by a procedure called *retiming*, developed by Leiserson and Saxe [5]. The core of the retiming procedure is in fact the one-dimensional version of the reindexing transformation discussed in the previous subsection. To each node $\mathbf{u}$ in the RG is assigned an integer value, $r_{\mathbf{u}}$, called a label. An edge which extends from node $\mathbf{u}$ to node $\mathbf{v}$ with weight $w_{\mathbf{uv}}$ in the RG will have weight

$$\overline{w}_{\mathbf{uv}} = w_{\mathbf{uv}} + r_{\mathbf{v}} - r_{\mathbf{u}} ,$$

after retiming. The labels represent relative *leads* or *lags* in clock values introduced to the computations denoted by the nodes of the RG. Originally, the computation of $\mathbf{v}$ lags behind that of $\mathbf{u}$ by $w_{\mathbf{uv}}$ cycles. After retiming, the lag is $\overline{w}_{\mathbf{uv}}$ cycles. A feasible retiming is a set of labels $\{r_{\mathbf{u}}\}$ such that all edges on the retimed RG have non-negative weights; a feasible retiming is optimal if the cycle time of the retimed RG is the minimum among all the feasible retimings. The relationship between retiming and reindexing is apparent: suppose τ is the linear schedule from which the RG is derived, then a retiming of RG with labels $\{r_{\mathbf{u}}\}$ is equivalent to a reindexing of the DG with offsets $\{\delta_{\mathbf{u}}\}$ where

$$r_{\mathbf{u}} \equiv \delta_{\mathbf{u}}\tau \qquad \forall\, \delta_{\mathbf{u}}\,.$$

Hence, knowing τ and the retiming which minimizes the cycle time, an appropriate reindexing can be determined.

Leiserson and Saxe have developed a polynomial time algorithm for finding an optimal retiming for a RG [5]. Note that if $\{r_{\mathbf{u}}\}$ is a feasible retiming then adding the same integer constant to the labels $\{r_{\mathbf{u}}\}$ gives another retiming which results in the same register assignments on the edges of the RG (and therefore, the same cycle time). Two such retimings are said to be *identical.* On the other hand, two non-identical retimings may also lead to the same cycle time; such retimings are called *equivalent.* The procedure given in [5] finds only one of the equivalent optimal retimings. In our implementation, we modify the procedure to find the optimal retiming which results in the minimum number of registers on the RG (and hence on the final implementation) [8].

Since retiming is a special case of reindexing, the total number of registers (l_j in (5)) in any of the elementary circuits in a RG is independent of retiming. We showed in [8] that if a RG can be retimed then there exists an optimal retiming such that all the edges which belong to none of the elementary circuits in the RG would have non-zero weights after retiming. The minimum cycle time of a RG under retiming, therefore, depends only on the number of registers in the elementary circuits. Thus, for our purpose, we can assume that the nodes in the DG (and hence, in the RG for any choice of τ) are either part of the elementary circuits or are isolated. This is an important observation since, together with the discussion in the previous subsection, it establishes that the values of both the number of systolic cycles, n_{sys}, and the cycle time, t_{sys}, depend only on the set of values of $\{l_j\}$, the number of registers in the elementary circuits in the RG.

In the next section, we shall show that the number of different $\{l_j\}$ that need to be considered is finite and a branch-and-bound method can be developed to determine the set of $\{l_j\}$ that minimizes the value of $n_{sys} \times t_{sys}$.

3. MAIN RESULTS

Given a set of URE's, the computation range, and the propagation delays of the functional units which compute the values of the variables, the objective is to find a linear schedule τ and a retiming, $\{r_{\mathbf{u}}\}$, such that the product $n_{sys} \times t_{sys}$ is minimized. First, a scheme for solving the problem with exhaustive enumeration is presented. A more efficient branch-and-bound method is then developed to handle the same problem.

Exhaustive enumeration

Suppose the DG contains m elementary circuits, $ec_1 \cdots ec_m$, where ec_j has λ_j edges and the sum of edge weights is S_j, $j = 1 \cdots m$. Since the DG and the RG have the same structure, the same scheme is used to name the elementary circuits in the RG.

Let $L = (l_1 \cdots l_m)$ be an integral m-vector with positive components, called a *register assignment.* Define the two functions *cycle_time* (L), and *ncycle* (L). The *cycle_time* function applies retiming to the RG, which is obtained from the given DG with l_j registers assigned to the jth elementary circuit, and returns the minimum cycle time and the set of labels found if a retiming exists, else it returns an infinite value as the cycle time,

$$\mathit{cycle\_time}\,(L) = \begin{cases} \text{min. cycle time} & \text{if retiming exists} \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

If *cycle_time* (L) is finite, the corresponding RG is called *retimeable.* The function *ncycle* determines the value of n_{sys} by solving an integer linear program,

$$ILP\,1\colon \begin{cases} \text{minimize} \quad 1 + Y \\ \text{subject to} \;\; Y - (v_i - v_j)\tau \geq 0, \;\; \forall\, v_i,\, v_j \in V \\ \qquad\qquad\;\; S_j \tau = l_j\,, \;\; j = 1 \cdots m \\ \qquad\qquad\;\; \tau \in \mathbf{Z}^n \end{cases}$$

where V is the set of vertices of the computation range (prior to reindexing). Let $\Delta r_{\max}$ be the maximum difference between any pair of labels returned from calling the function *cycle_time* (L). The value returned by *ncycle* then equals $(Y + \Delta r_{\max} + 1)$ if *ILP1* has a solution, otherwise, it returns an infinite value. Note that if *ncycle* (L) is finite then *cycle_time* (L) must be finite; the converse is not true, however.

With these two functions, the optimal choice of L which minimizes the value of $n_{sys} \times t_{sys}$ can be determined from exhaustive enumeration if, for all j, the number of l_j's that must be considered is bounded.

Bound on search space

An elementary circuit ec_j in the RG is said to be *saturated* if there are at least λ_j registers in the circuit, where λ_j is the number of edges in ec_j, else it is said to be *unsaturated*. An edge is said to be *free* if it is shared only by saturated elementary circuits, otherwise, it is said to be *bound*.

Claim : If a RG is retimeable then there exists an optimal retiming such that all the free edges in the retimed RG have non-zero edge weights.

Proof : Given in [8].

With this claim, all the free edges in the saturated elementary circuits can be assumed to have non-zero weights after retiming. Since the cycle time of a RG is determined by the zero-paths in the graph, as far as finding the minimum cycle time is concerned, all the free edges in the RG can be ignored (because they have non-zero edge weights). Thus, the value of $cycle\_time(L)$ depends only on the number of registers in the unsaturated elementary circuits in the RG, that is, the set of values $\{ l_j \mid l_j < \lambda_j \}$. Hence, it suffices to consider only the set of register assignments $\{ L \mid 1 \le l_j \le \lambda_j, \ j = 1 \cdots m \}$ if the functions $cycle\_time$ and $ncycle$ are redefined in the following manner:

> Without loss of generality, suppose L is such that the first i elementary circuits, $ec_1 \cdots ec_i$ are saturated and the rest of the elementary circuits are unsaturated, that is, $L = (l_1 \cdots l_i \ l_{i+1} \cdots l_m)$ where $l_j = \lambda_j$, $j = 1 \cdots i$, and $l_j < \lambda_j$, $j = i+1 \cdots m$. Then, the value of $cycle\_time(L)$ is given by
>
> $$cycle\_time(L) = \begin{cases} \cdot \text{ delete all the edges which are not part of any of } ec_{i+1} \cdots ec_m \text{ in the DG} \\ \cdot \text{ form the RG : assign } l_j \text{ registers to } ec_j, \ j = i+1 \cdots m \\ \cdot \text{ if optimal retiming for the RG exists then return cycle time else return } \infty \end{cases},$$
>
> and the value of $ncycle(L)$ is $(1 + Y + \Delta r_{\max})$ where Y is obtained from solving the ILP
>
> $$ILP\,2: \begin{cases} \text{minimize} & 1 + Y \\ \text{subject to} & Y - (v_i - v_j)\tau \ge 0, \ \forall \, v_i, v_j \in V \\ & S_j \tau \ge \lambda_j, \ j = 1 \cdots i \\ & S_j \tau = l_j, \ j = i+1 \cdots m \\ & \tau \in \mathbf{Z}^n \end{cases}.$$
>
> The constraints $S_j \tau \ge \lambda_j$, $j = 1 \cdots i$, in *ILP2* ensure that the elementary circuits $ec_1 \cdots ec_i$ are saturated.

The size of the search space for finding the register assignment L that minimizes the product $n_{sys} \times t_{sys}$ is therefore $\prod_{j=1}^{m} \lambda_j$.

Branch-and-bound method

In this subsection, we utilize a decision tree of m levels to construct the optimal L vector *incrementally*. A simple branch-and-bound method is used to prune the decision tree to improve the efficiency. The performance of this approach depends on the quality of the initial solution and the lower bounding function.

Each level of the decision tree represents one of the components of L : the first level contains λ_1 nodes with labels 1, $\cdots$, λ_1–1, and λ_1^+, where λ_1^+ denotes an integer value greater than or equal to λ_1; for a node with label l_1 at the first level, there are λ_2 nodes at the second level with labels $l_1 1$ to $l_1 \lambda_2^+$, and so on. Nodes at the ith level of the decision tree have labels of length i, $l_1 l_2 \cdots l_i$, and therefore, the labels of the leaf nodes on the tree correspond to the full L vectors.

Let L_i be the label of a node, $node_i$, at the ith level. Define the two functions $lb\_cycle\_time(L_i)$ and $lb\_ncycle(L_i)$ as

$$lb\_cycle\_time\,(L_i) = \begin{cases} \cdot\ \text{delete all the edges which are not part of any of } ec_1 \cdots ec_i \\ \cdot\ \text{delete all edges shared only by } \{ec_j \mid l_j = \lambda_j^+ ,\ j = 1 \cdots i\} \\ \cdot\ \text{form the RG} \\ \cdot\ \text{if optimal retiming for the RG exists then return cycle time else return } \infty \end{cases}$$

$$lb\_ncycle\,(L_i) = \begin{cases} \text{minimize} \quad 1 + Y \\ \text{subject to } \ Y - (v_i - v_j)\tau \geq 0, \quad \forall\, v_i, v_j \in V \\ \qquad S_j\tau = l_j \quad \text{if } l_j < \lambda_j ,\ j = 1 \cdots i \quad (\text{unsaturated } ec\,) \\ \qquad S_j\tau \geq \lambda_j \quad \text{if } l_j = \lambda_j^+ ,\ j = 1 \cdots i \quad (\text{saturated } ec\,) \\ \qquad S_j\tau \geq 1 ,\ j = i+1 \cdots m \quad (l_j \text{ not yet assigned}) \\ \qquad \tau \in \mathbf{Q}^n \end{cases} .$$

Clearly, any of the leaf nodes in the subtree that roots at $node_i$ has label L of the form $(L_i \mid l_{i+1} \cdots l_m)$ for some $(l_{i+1} \cdots l_m)$, and

$$\begin{cases} lb\_cycle\_time\,(L_i) \leq cycle\_time\,(L\,) \\ lb\_ncycle\,(L_i) \leq ncycle\,(L\,) \end{cases} .$$

Thus, if the value of $lb\_ncycle\,(L_i) \times lb\_cycle\_time\,(L_i)$ is already larger than that of the incumbent solution then the subtree at $node_i$ should be pruned. This provides us a bounding function for the branch-and-bound method:

$$lb\,(L_i) = lb\_ncycle\,(L_i) \times lb\_cycle\_time\,(L_i)\ .$$

The computation cost of the bounding function is low (polynomial time) : $lb\_ncycle\,(L_i)$ is a linear program which can be solved with the Simplex method and $lb\_cycle\_time\,(L_i)$ can be solved with the polynomial time algorithm given in [5]. At the leaf level, however, the lb_ncycle function has to be replaced by the $ncycle$ function to produce an integral solution τ. Due to the interaction among elementary circuits which share part of their edges, for many of the partial assignments, L_i, either no retiming exists or the constraint set of the linear program in $lb\_ncycle\,(L_i)$ is inconsistent. The subtree that roots at a node with such a label L_i can be pruned. To take advantage of this property, elementary circuits which share edges should, if possible, be numbered consecutively.

For our problem, a reasonable initial solution can be taken as the one which minimizes one of the two objectives: n_{sys} and t_{sys}. The first choice could result in sequential evaluation of each of the variables in the DG and is therefore related to the single functional unit per processor model discussed in Section 1. The second choice may require all the functional units represented by nodes in the RG to be active in parallel and hence, the number of functional units per processor may have to match the number of nodes in the RG.

An example

To illustrate the tradeoff between the values of n_{sys} and t_{sys}, consider one of the steps in the Toeplitz Hyperbolic Cholesky Solver [7], which can be expressed in a set of six URE's :

$$\begin{aligned} \rho_{j+1,j} &= \mathbf{s}_{j+1,j-1}/\mathbf{r}_{j,j-1} & \qquad \mathbf{s}_{ij} &= \mathbf{s}_{i,j-1} - \mathbf{q}_{ij} \\ \mathbf{r}_{ij} &= \mathbf{r}_{i-1,j-1} - \mathbf{p}_{ij} & \qquad \mathbf{q}_{ij} &= \overline{\rho}_{ij} \times \mathbf{r}_{i-1,j-1} \\ \mathbf{p}_{ij} &= \overline{\rho}_{ij} \times \mathbf{s}_{i,j-1} & \qquad \overline{\rho}_{ij} &= \begin{cases} \overline{\rho}_{i-1,j} & \text{if } i \neq j+1 \\ \rho_{ij} & \text{otherwise} \end{cases} \end{aligned}$$

The range of computation is a triangle, defined by the inequalities $1 \leq j < i \leq N$, where N is the size of the Toeplitz matrix. The DG of this set of URE's is shown in Fig. 4a. Let the propagation delays of the register, adder, multiplier and divider be 0, 1, 5, and 5 time units respectively.

Consider first the n_{sys}-optimal design where $\tau = (1 \ \ 1)^{\mathrm{T}}$ and $n_{sys} = 2N-2$. The corresponding RG for this choice of τ is depicted in Fig. 4b. The minimum cycle time is $t_{sys} = 11$. Hence, the overall time performance of the systolic implementation using $\tau = (1 \ \ 1)^{\mathrm{T}}$ for this set of URE's is $n_{sys} \times t_{sys} = (2N-2) \times 11$ time units. For $N = 10$, say, the value equals 198 time units.

On the other hand, a t_{sys}-optimal design can be obtained from picking, say, $\tau = (1 \ \ 3)^{\mathrm{T}}$. This choice increases the value of n_{sys} to $4N-1$ but, with retiming, the value of t_{sys} is reduced to 5 time units (see Fig. 5a and 5b), which is the minimum achievable. The overall time performance is therefore, $(4N - 1) \times 5$ time units. For $N = 10$, the total time required equals 195 time units which is marginally better than that of the n_{sys}-optimal design.

Now, consider the linear schedule, $\tau = (1 \ \ 2)^T$. The corresponding RG is shown in Fig. 6a. The value of n_{sys} is $3N$, and the value of t_{sys} obtained from retiming (Fig. 6b) is 6, neither of which is the minimum value achievable. However, the overall time performance of the schedule equals $3N \times 6$, which is better than the previous two schedules for $N > 5$. Implementation using this (optimal) linear schedule requires roughly 10% less time than the previous two designs to compute the same set of URE's. This improvement in performance is essentially independent of the actual value of N, provided that N is larger than 10. □

4. PERFORMANCE EVALUATION

Due to space limitation, implementation details of the branch-and-bound method are omitted. The effectiveness of the branch-and-bound method can be assessed by the percentage or the number of nodes on the decision tree it traverses. As mentioned in the previous section, at a non-leaf node of the decision tree the bounding function can normally be computed in polynomial time. At a leaf node, however, one needs to solve an integer linear program (the *ncycle* function) which is computationally expensive. Hence, an alternative measure of performance of the branch-and-bound method is the number of leaf nodes that survive the pruning.

Tabulated below are preliminary statistics obtained from applying the branch-and-bound method to two examples: the Toeplitz Hyperbolic Cholesky Solver (THCS) shown in the previous section, and the dynamic programming algorithm (DP). Details of the algorithms are immaterial, only those characteristics which are relevant to our discussion are provided below:

Algorithm	Dependence Graph (DG)			Decision Tree		
	No. of vertices	No. of $ec's$	No. of edges in ec_i	Tree size	No. of nodes traversed	No. of surviving leaves
THCS	6	5	4,6,6,4,4	3052	84 (2.75%)	3
DP	9	4	4,3,4,3	208	17 (8.17%)	1

We quoted only the number of elementary circuits which have more than one node (*i.e.*, the non self-loops) in the above table. As can be noted, the branch-and-bound scheme works extremely well in pruning the decision trees. For decision trees of small size (*e.g.*, DP above), however, the percentage of nodes traversed may be large although the *actual* number of nodes traversed will remain small. The execution time of the branch-and-bound method for the above two examples are respectively 14.7 and 7.4 CPU seconds on a SUN/3 workstation running BSD 4.3 Unix operating system. These runtimes do not include the time for solving the integer linear programs at the leaf nodes.

5. CONCLUDING REMARKS

In this paper, the derivation of time-optimal linear schedules for algorithms that can be expressed in URE's is formulated as a combinatorial optimization problem. We showed that the solution space of the problem is bounded and proposed an efficient branch-and-bound method for its solution. The method applies also to the case when the processors on the arrays are assumed to contain pipelined functional units — the so called two-level pipeline implementations of systolic arrays, a well-known example of which is the WARP computer developed at CMU [9]. Refer to [8] for details.

In the discussion, we assumed that the processors of the array have enough hardware resources (functional units) to support the required level of parallelism. While this assumption is valid for VLSI/WSI implementations of systolic arrays, it does not necessarily apply to general purpose, fixed architecture systolic machines. A major difficulty in generalizing the branch-and-bound method to solve the optimization problem with fixed hardware resources is that the optimal retiming for a RG can no longer be determined in polynomial time. In fact, to compute just the cycle time of a given RG for a feasible retiming under hardware constraint is equivalent to solving a multiprocessor scheduling problem (remove all the non-zero edges from the retimed RG and treat the resultant graph as a task precedence graph where the propagation delay associated to a node is taken as the task processing time) which is known to be *NP−hard*. Although good heuristics exist for handling the multiprocessor scheduling problem [10], the number of iterations required to determine the optimal retiming is still exponential since all possible arrangements (shiftings) of registers on the RG have to be explored. Further research is needed to develop an efficient heuristic method for determining a close-to-optimal retiming in the presence of hardware constraints.

Acknowledgement The work presented in this paper was supported by the Army Research Office (ARO) under contract number DAAL03-86-K-0158 and by the Office of Naval Research (ONR) under contract number N00014-85-K-0461.

References

[1] Kung, H. T., and Leiserson, C. E., "Systolic Arrays (for VLSI)," in *Sparse Matrix Proc. 1978*, I. S. Duff and G. W. Stewart, eds., SIAM, 1978.

[2] Kung, H. T., "The Structure of Parallel Algorithms," in *Advances in Computers*, Vol. 19, pp. 65-112, Academic Press, 1980.

[3] Fortes, J. A. B., Fu, K. S., and Wah, B. W., "Systematic Approaches to the Design of Algorithmically Specified Systolic Arrays," *Proc. IEEE Conf. on ASSP*, pp. 300-303, 1985.

[4] Karp, R. M., Miller, R. E., and Winograd S., "The Organization of Computations for Uniform Recurrence Equations," *JACM*, Vol. 14, No. 3, pp. 563-590, 1967.

[5] Leiserson, C. E., and Saxe, J. B., "Optimizing Synchronous Systems," *Proc. 23rd Symp. on Foundations of Computer Science*, pp. 23-36, Oct. 1981.

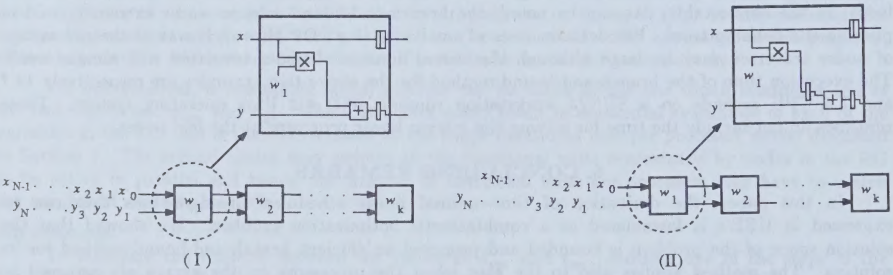
[6] Papadimitriou, C. H., and Steiglitz, K., *Combinatorial Optimization Algorithms and Complexity*, Prentice-Hall, Inc., 1982.

[7] Delosme, J.-M., and Ipsen, I.C.F., "Systolic Array Synthesis: Computability and Time Cones," *Research Report No. RR-474*, Dept. of Computer Science, Yale University, May 1986.

[8] Wong, Y., and Delosme, J.-M., "Optimization of Computation Time for Systolic Arrays," *Research Report No. RR-651*, Dept. of Computer Science, Yale University, Sept. 1988.

[9] Kung, H. T., "Systolic Algorithms for the CMU Warp Processor," in *Systolic Signal Processing*, E. E. Swartzlander, ed., pp. 73-96, Marcel Dekker, 1987.

[10] Kasahara, H., and Narita, S., "Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing," *IEEE Trans. on Computers*, Vol. C-33, No. 11, pp. 1023-1029, Nov. 1984.



x
y
w_1
$x_{N-1} \cdots x_2\, x_1\, x_0$
$y_N \cdots y_3\, y_2\, y_1$
w_1
w_2
w_k
(I)
(II)

Fig. 1: Two systolic arrays for computing the one-dimensional convolution. The small rectangular boxes inside the processors denote register elements.

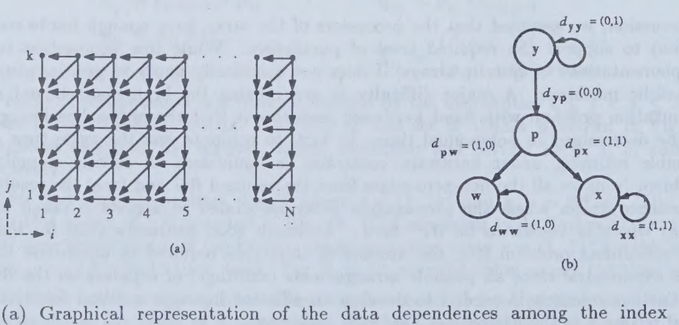


Fig. 2: (a) Graphical representation of the data dependences among the index points in the computation range ($k = 5$), and (b) the corresponding dependence graph (DG).

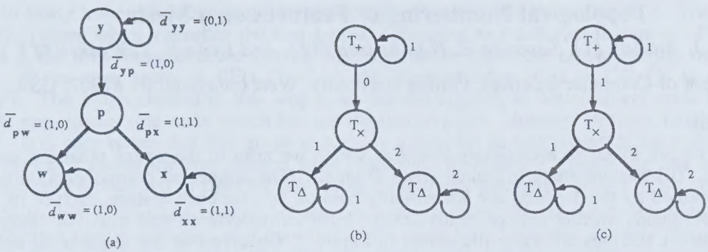

y
d̄yy = (0,1)
d̄yp = (1,0)
p
d̄pw = (1,0)
d̄px = (1,1)
w
x
d̄ww = (1,0)
d̄xx = (1,1)
(a)
T+
1
0
T×
1
2
TΔ
1
TΔ
2
(b)
T+
1
1
T×
1
2
TΔ
1
TΔ
2
(c)

Fig. 3: (a) DG of the one-dimensional convolution algorithm after reindexing; (b) the corresponding RG's before and (c) after reindexing.

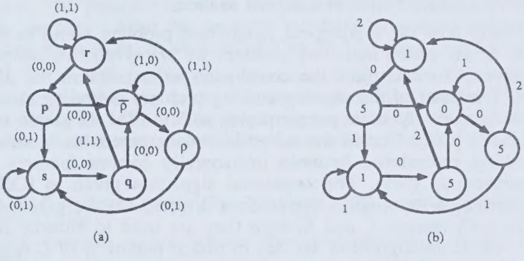

(1,1)
r
(0,0)
(1,0)
(1,1)
p
(0,0)
p̄
(0,0)
(0,1)
(1,1)
(0,0)
ρ
(0,0)
s
q
(0,1)
(0,1)
(a)
2
1
0
1
2
5
0
0
0
1
2
0
5
0
1
5
1
1
(b)

Fig. 4: (a) DG of one of the steps in THCS and (b) the corresponding RG with $\tau = (1\ \ 1)^{\mathrm{T}}$.

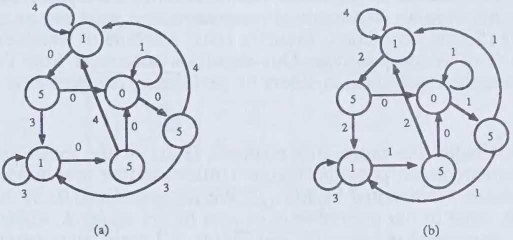

4
1
0
1
4
5
0
0
0
3
4
0
5
0
1
5
3
3
(a)
4
1
2
1
1
5
0
0
1
2
2
0
5
1
1
5
3
1
(b)

Fig. 5: (a) RG of THCS with $\tau = (1\ \ 3)^{\mathrm{T}}$ before and (b) after retiming.

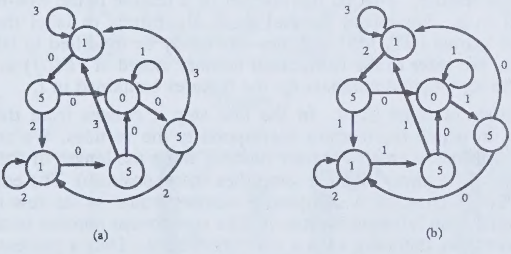

3
1
0
1
3
5
0
0
0
2
3
0
5
0
1
5
2
2
(a)
3
1
2
1
0
5
0
0
1
1
1
0
5
1
1
5
2
0
(b)

Fig. 6: (a) RG of THCS with $\tau = (1\ \ 2)^{\mathrm{T}}$ before and (b) after retiming.

Topological Numbering of Features on a Mesh

Mikhail J. Atallah (†), Susanne E. Hambrusch (††), and Lynn E. TeWinkel(†††)

Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA.

I. Introduction

Assume we are given an $n \times n$ binary image in which we refer to the image positions containing a '1' (resp. '0') as containing a 1-pixel (resp. 0-pixel). The connected components, which we call *features*, formed by the 1-pixels are *horizontally convex*; i.e., for each feature, each of its row's 1-pixels are contiguous on that row or, equivalently, form an interval on that row. An example of horizontally convex features are those illustrated in Figure 2. Observe that the features do not need to be vertically convex. For horizontally convex features the relationship "to the left of" defines a partial order as follows. A feature f is *to the left* of a feature g, denoted $f \prec g$, if and only if a 1-pixel of f is to the left of a 1-pixel of g on some row of the image. The *topological numbering problem* is to assign to every feature in the image a unique number so that for any feature f, all features to the left of f have a smaller number assigned to them.

We next explain briefly how the topological numbering problem arises in the context of the stereo matching problem. In the stereo matching problem we are given two images, L and R, of a 3D scene and the objective is to determine the coordinates of a point in the 3D scene from its image points in L and R. The heart of the stereo matching problem is finding corresponding points in the images L and R. An elegant dynamic programming solution for the stereo matching problem is given by Ohta and Kanade [OK]. One of the subproblems that needs to be solved in their algorithm is that of establishing precedence between horizontally convex features of an image by assigning topological numbers to them. The sequential algorithm given in [OK] for numbering features is based on performing a topological search on a directed acyclic graph. If the topological numbers are obtained for both images L and R, then they are used to enforce consistency in the matching process as follows. If the algorithm decides to match feature f_l of L with feature f_r of R, then all features in L with a smaller number than f_l can only be matched with features in R that have a smaller number than f_r.

In the next section we describe an algorithm that determines the topological numbers in $O(n)$ time when the image is stored in an $n \times n$ mesh of processors, one pixel per processor. We assume the mesh to be an SIMD (Single Instruction, Multiple Data) machine in which every processor has a fixed (i.e. $O(1)$) amount of of local storage. Our algorithm constructs from the image a rooted, directed in-tree and obtains the topological numbers by performing computations on that tree.

II. The Algorithm

Throughout we will index the processing elements (PEs) of the mesh by either their row-major index or by their row/column position. Figure 1 illustrates the row-major indexing scheme. The PE at row i and column j is denoted by PE(i,j). We let position (0,0) of the mesh be the top left PE of the mesh. The input to our algorithm is an $n \times n$ binary image I, where $I(i,j)$ is stored at PE(i,j). Our algorithm assumes that every PE containing a 1-pixel also contains the connected component number of the corresponding feature. We furthermore assume that the component numbers are assigned so that the number of a feature is the largest row-major index of any PE containing a 1-pixel from that feature. That is, the number of a feature is the row-major index of the lowest rightmost 1-pixel in it. Numerous parallel mesh algorithms to label the connected components in $O(n)$ time are known [HT, MS] and they can easily be modified to label the features as needed by our algorithm. We refer to the component number stored at PE(i,j) as $C(i,j)$. The output of our algorithm is the topological numbers for the features contained in I.

Our algorithm consists of three steps. In the first step it creates from the input image I a rooted, directed in-tree T in which the features correspond to the vertices. We create a root vertex which corresponds to an artificially created feature running along the length of the left border of the input image (introducing this "dummy" feature simplifies the exposition). The edges of the tree are created as follows. If PE(i,j) contains a component number equal to its row-major index, then PE(i,j) initiates a horizontal scan leftward on row i. The component number is taken along on this scan. In Figure 2, the processors initiating such a scan are circled. Thus a processor initiates a scan if and only if it contains the lowest rightmost 1-pixel of a feature. From now on we will refer to the largest row number (i.e. geometrically lowest row) containing 1-pixels belonging to feature f as low(f). For example, in Figure 2, low(12)=1, and low(47)=4. If feature f contains more than one 1-

---

† Work supported by ONR Grants N00014-84-K-0502 and N00014-86-K-0689, and by NSF Grant DCR-8451393, with matching funds from AT&T. Part of this work was done while visiting RIACS, at NASA Ames Research Center.

†† Work supported by ONR Contracts N00014-84-K-0502 and N00014-86K-0689, and by NSF Grant MIP-87-15652.

††† Work supported by ONR Contract N00014-86K-0689.

pixel in low(f), then the scan may travel through other 1-pixels belonging to f. The scan initiated at PE(i,j) stops when it reaches the first 1-pixel belonging to a different feature g. For example, in Figure 2, the scan initiated at the circled '85' stops at the tip of the arrow starting from it (i.e., at a '90'). If the scan stops at PE(i,j'), $j'<j$, the directed edge ($C(i,j)$, $C(i,j')$) is created in the PE(i,j'). The graph created in this step is an acyclic digraph in which every node has out-degree exactly one, except one node which has out-degree zero (the "dummy" feature, labeled '90' in Figure 2). It is easy to see that this graph is actually a directed in-tree rooted at the node of out-degree zero. Figure 3 shows the tree created from Figure 2. In Figure 3, notice that the children of any node are drawn from left to right by increasing component numbers. Thus T is an ordered tree.

Note that T does not contain all the information about the partial order. Figure 4 illustrates three distinct partial orderings that give rise to the same tree. However, the tree created in our algorithm contains all of the crucial information needed to compute the topological numbers we seek.

The second step of the algorithm takes the rooted, directed in-tree T and computes the preorder numbers of the vertices of T using the algorithm given in [AH]. When computing the preorder numbers it is crucial that the children of every node are visited in the order of increasing component numbers. The preorder numbers for the tree shown in Figure 3 are given inside the circles indicating the vertices. After the preorder numbers have been assigned, the final step of the algorithm broadcasts them from the vertices of T to all 1-pixels of the corresponding features in I.

Even though the tree T created does not contain all of the information about the "$\prec$"partial ordering, its preorder numbers are the correct topological numbers for "$\prec$". The proof of this nontrivial fact can be found in [AHT]. The implementation of the 3 steps of the algorithm in $O(n)$ time is straightforward and accomplished using standard methods.

References

[AH] M.J. Atallah and S.E. Hambrusch, "Solving Tree Problems on a Mesh-Connected Processor Array", *Information and Control*, Vol. 69, pp. 168-187, 1986.

[AHT] M.J. Atallah, S.E. Hambrusch, and L.E. TeWinkel, "Topological Numbering of Features on a Mesh", CSD-TR-779, Purdue University, 1988.

[HT] S.E. Hambrusch and L.E. TeWinkel, "A Study of Connected Component Labeling Algorithms on the MPP", *Proceedings of Third International Conference on Supercomputing*, Vol. I, pp. 477-483, 1988.

[MS] R. Miller and Q.F. Stout, *Parallel Algorithms for Regular Architectures*, manuscript, 1988 (to be published by MIT Press).

[OK] Y. Ohta and T. Kanade, "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming", *IEEE Transactions on PAMI*, Vol PAMI-7, pp. 139-154, 1985.

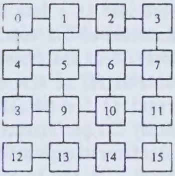


Figure 1

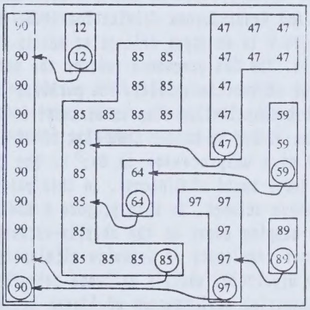


Figure 2

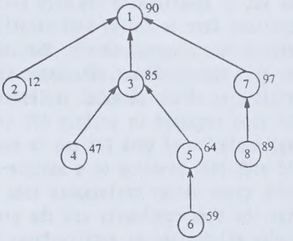


Figure 3

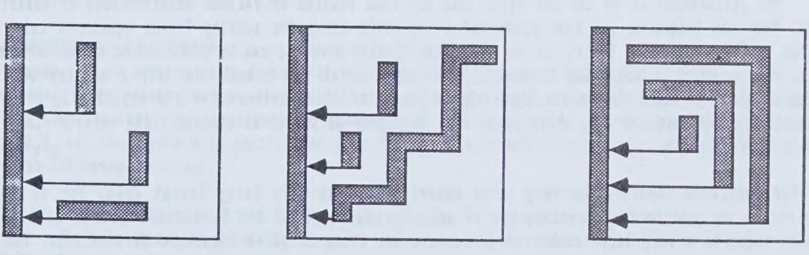


Figure 4

SOLUTION OF LINEAR EQUATIONS ON SHUFFLE-EXCHANGE AND MODIFIED SHUFFLE EXCHANGE NETWORKS

R.B.Panwar* and L.M.Patnaik**

* Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore, 560012, India

** Department of Computer Science and Automation
Indian Institute of Science, Bangalore, 560012, India

ABSTRACT

In this paper we present parallel implementations of algorithms for solving linear equations of the form **Ax=b**, where **A** is an n x n full, unstructured, nonsingular matrix. We use the Gauss-Jordan elimination algorithm (GJE) with pivoting since a parallel implementation of GJE gives better processor utilization than that of the Gaussian Elimination method (GE). The processors are assumed to be interconnected as a shuffle-exchange network. A modified shuffle-exchange interconnection scheme is presented on which the mapping of the GJE algorithm is shown to have better performance than that on the shuffle-exchange network. The parallel implementation of GJE requires $\Omega(n\log n)$ time. We show that GJE can be performed using the shuffle-exchange interconnection scheme in $O(n\log^2 n)$ time using $n^2/\log^2 n$ processors (PEs). It is shown that GJE can also be performed using the shuffle-exchange interconnection scheme in $O(n\log n)$ time using n^2 processors. This performance is not optimal with respect to the product of the number of processors used and the time required. On the proposed modified shuffle-exchange network, GJE can be performed in $O(n\log n)$ time using $n^2/\log n$ PEs. It is shown that the modified shuffle-exchange interconnection requires a layout area of $O(N^2/\log N)$ for an N PE configuration.

1. INTRODUCTION Most scientific problems involve solution of linear equations of the form Ax=b where A is a square matrix. The methods commonly used to solve dense, unstructured systems of linear equations are Gaussian Elimination (GE) and Gauss-Jordan Elimination (GJE) [3]. The GE procedure reduces the set of equations to the form Ux=c where U is an upper triangular matrix. The solution of the system of equations Ux=c is called back-substitution. The GJE procedure reduces the set of equations to a form Dx=c where D is a diagonal matrix. The solution of Dx=c is trivial. The parallel implementation of GJE requires the same time as GE but eliminates the back-substitution step encountered in the latter. Thus the processor utilization of the parallel implementation of GJE is better than that of GE [4]. For a matrix of size nxn, the time required to perform GJE or GE on a uniprocessor is $O(n^3)$. The time required for parallel implementation of both GJE and GE has a lower bound of $\Omega(n\log n)$. In this paper we study the performance of GJE with full pivoting on a shuffle-exchange network. We also propose a modified shuffle-exchange network which gives better performance than the mapping shown on the shuffle-exchange network. (It may be noted that the time complexity and the processor complexity of Gaussian elimination is same as that of Gauss-Jordan elimination on architectures like hypercube, shuffle exchange network or cube-connected cycles. We consider GJE only as a case study for comparing performances of linear equation solution algorithms with pivoting on various architectures.)

The performance of GE and GJE algorithms has been studied on various architectures by different authors. Kung and Leiserson [6] have presented a systolic array for solving linear equations using GE algorithm without pivoting. In [1], it is shown that GE with pivoting can be performed on an nxn matrix in $O(n^{5/3})$ time using $n^{4/3}$ Processing Elements (PEs) connected as a 2-D mesh. In [9], a mapping of the Gaussian elimination algorithm on the hypercube is presented which solves an n x n equation in $O(n\log n)$ time using n^2 processors. In [2], it is shown that GE can be performed in $O(n\log n)$ time using $n^2/\log n$ PEs connected as a hypercube.

The hypercube topology has very large connectivity and very large layout area. For an N PE configuration we need logN connections per PE and a layout area of $O(N^2)$. Since solution of linear equations requires a very large number of processors (as shown in [2] a hypercube of $O(n^2/\log n)$ PEs is required for an n x n matrix) we need to perform GE (or GJE) on a topology which has lower connectivity and lesser layout area than hypercube. In [7] we have shown that Gauss-Jordan elimination can be performed in $O(n\log n)$ time using $n^2/\log n$ processors connected as cube-connected cycles. The cube-connected cycles topology has an $O(N^2/\log^2 N)$ area for an N processor configuration with atmost three connections per processor [8]. In this paper we show that GJE can be performed in $O(n\log^2 n)$ time using $n^2/\log^2 n$ PEs connected as a shuffle-exchange network. It is also shown that GJE can be performed in $O(n\log n)$ time using n^2 PEs connected as shuffle-exchange [10]. The shuffle-exchange network has atmost three connections per PE and a layout area of $O(N^2/\log^2 N)$ for an N PE configuration. We also propose a modified shuffle-exchange network (described in section 4) on which GJE can be performed in $O(n\log n)$ time using $n^2/\log n$ processors. This performance is same as that of the hypercube [2] or cube-connected cycles [7] for solution of linear equations. The modified shuffle exchange has a constant connectivity of six per PE. A layout area of the modified shuffle-exchange network which requires $O(N^2/\log N)$ area for an N PE configuration is shown. The parallel architectures considered above work in an SIMD fashion.

2. GAUSS-JORDAN ELIMINATION ALGORITHM

The Gauss-Jordan elimination algorithm is presented in algorithm 1 below.

Algorithm 1

```
begin
   for k:= 1 to n do
  /* eliminate the terms of the kth column other than $a_{kk}$ */
    begin
      for l:= k to n do
      /* find pivot k */
          find m such that $|a_{mk}| = \max |a_{lk}|$ ;

      interchange ROW[m] and ROW[k];
      for all i <> k, j $\geq$ k do
          $a_{ij} \leftarrow a_{ij} - a_{ik} * a_{kj} / a_{kk}$ ;
    end;
   /* the matrix A is reduced to diagonal matrix D */
   solve the set of equations Dx=c ;
end.
```

The algorithm reduces the given matrix A to a diagonal matrix D such that the solution of the resulting diagonal set of equations becomes very simple. The time required for the sequential execution of algorithm 1 is $O(n^3)$. The solution of the set of equations given by Ax=b where A is a diagonal matrix is given in algorithm 2.

Algorithm 2

```
begin
   for k:=1 to n do
        $x_k \leftarrow b_k/a_{kk}$;
end.
```

The time required for algorithm 2 is $O(n)$.

3. PERFORMING GJE IN PARALLEL

Assume that we have to solve the set of equations **Ax=b**. We perform GJE on the augmented matrix **[A|b]** of size nx(n+1). We assume that we perform GJE on a matrix **A** of size nxn since this does not affect the overall time complexity of the algorithm. Asssume that the total number of processors used to perform GJE is $n^2/k_r k_c$ and the matrix A is partitioned into $n^2/k_r k_c$ submatrices of size $k_r \times k_c$ each. Further discussion uses the following notations.

A[i,j] : the (i,j)th element of the matrix A, $0 \leq i,j < n$,
ROW[q] : The qth row of the matrix A.
COL[q] : The qth column of the matrix A.

PA[r,c] : the (r,c)th partition of the matrix A, $0 \le r < n/k_r$, $0 \le c < n/k_c$: thus PA[r,c] is the set of the elements A[i,j], $r.k_r \le i \le ((r+1).k_r-1)$, $c_c k \le j \le ((c+1)k_c-1)$,

PR[q] : The qth row of partitions which is same as the set of partitions, PA[q,c], for all c=0 to n/k_c-1,

PC[q] : The qth column of partitions which is same as the set of partitions, PA[r,q], for all r=0 to n/k_r-1,

PE-R[q] : the set of PEs containing the partitions belonging to PR[q], (thus all the elements of matrix A belonging to the set of partitions PR[q] are present in the set of PEs denoted by PE-R[q]).

PE-C[q] : the set of PEs containing the partitions belonging to PC[q]. (similarly)

According to the above notation, the elements of the ROW[q] (COL[q]) are mapped onto all the PEs belonging to the set of PEs, PE-R[q div k_r] (PE-C[q div k_c]), where the div operation represents integer division. We define the following operations on the elements of the matrix A.

1) COMMUNICATE-PIVOT(q) : This operation requires the communication of the pivot element present in one PE to all PEs.

2) FIND-PIVOT(q) : Assuming complete pivoting this operation corresponds to finding the maximum of all the elements A[i,j], $i \ge q$, $j \ge q$.

3) COMMUNICATE-ROW(q) : The elements of a ROW present in one PE-R are communicated to all other PE-Rs. Thus a row is communicated to all other rows.

(The elements of the ROW[q], present in the set of the PEs denoted by PE-R[q div k_r] are communicated such that all the elements of ROW[q] belonging to the partition PA[q div k_r, j], $0 \le j < n/k_c$ are communicated to all the PEs belonging to PE-C[j], for all j, $0 \le j < n/k_c$).

4) COMMUNICATE-COL(q) : The elements of a COL present in one PE-C are communicated to all other PE-Cs. Thus a column is communicated to all other columns.

Using the above operations, GJE can be performed as given in algorithm 3. The GJE algorithm is expressed in terms of the ELIMINATE-COLUMN operation described in algorithm 4.

Algorithm 3

```
 begin
     input the matrix A;
     for q:=1 to n do  ELIMINATE-COLUMN(q);
 end;
```

The ELIMINATE-COLUMN(q) operation eliminates the elements of the qth column as shown in algorithm 4.

Algorithm 4

```
ELIMINATE-COL(q)
 begin
    FIND-PIVOT(q);
   /* find the maximum of the elements A[i,j], i≥q and j≥q */
   /* assume that the maximum element is A[i',j'] */
    interchange ROW[q] and ROW[i'];
    interchange COL[q] and COL[j'];
    COMMUNICATE-PIVOT(q);
    COMMUNICATE-ROW(q);
    COMMUNICATE-COL(q);
    UPDATE;
 end;
```

After the COMMUNICATE-COL(q) operation in the algorithm 5, every PE has all the elements of the matrix A required to update the elements of the partition allocated to it i.e. the element a_{ij} will have the elements a_{iq}, a_{qj} and a_{qq} required to update it. The UPDATE step corresponds to performing the following operation on the elements of the matrix A.

if i<>q and $j \ge q$ then A[i,j]:=A[i,j]-A[i,q]*A[q,j]/A[q,q].

4. GJE ON THE SHUFFLE-EXCHANGE NETWORK

The shuffle-exchange interconnection is based on two interconnections i.e. the shuffle interconnection and the exchange interconnection.

a) **Shuffle Interconnection** : In the shuffle interconnection [10] the PE[i] is connected to the PE whose address is obtained by cyclically rotating the bit representation of i to the left by one bit. Thus $PE[(a_{(m-1)} \dots a_1 a_0)]$ is connected to $PE[a_{(m-2)} \dots a_1 a_0 a_{(m-1)}]$ by a unidirectional link (fig. 1).

b) **Exchange Interconnection** : In the exchange interconnection every $PE[a_{(m-1)} \dots a_1 a_0]$ is connected by a unidirectional link to $PE[a_{(m-1)} a_{(m-2)} \dots a_1 a'_0]$.

Here a'_0 denotes the binary complement of a_0.

Below we describe certain properties of the shuffle-exchange interconnection useful for the analysis of the performance of GJE.

A. Properties of Shuffle-Exchange Interconnection

Let the shuffle-exchange network consist of $n=2^m$ PEs. Also assume that every PE[i] contains a data element denoted by e_i. In the following discussion, performing the shuffle operation on the data element present in $PE[a_{(m-1)} \dots a_1 a_0]$ is equivalent to communicating the data element present in $PE[a_{(m-1)} \dots a_1 a_0]$ to $PE[a_{(m-2)} \dots a_1 a_0 a_{(m-1)}]$.

The following proposition is given in [6] and hence the proof is not given.

Proposition 1 Let 2^m PEs be connected as a shuffle-exchange network. Consider the data elements present in the pairs of processors PE[i] and PE[j], where the bit representations of i and j differ in only the position **(m-k)** for some **k**, $1 \le k \le m$. After **k** shuffles these data items are located in adjacent PEs.

Proposition 2 Let **N** PEs be connected as a shuffle-exchange network, such that $N=2^m$. Let the data element e_i be present in the PE[i] for some i, $0 \le i < N$. Then any semigroup operation e.g. maximum, minimum, sum or product computation can be performed on all the elements e_i in time O(m).

Proof The following is a constructive proof to perform the semigroup computation in **O(m)** time. Consider the elements e_i and e_j present in PE[i] and PE[j] respectively, such that the bit representations of i and j differ only in the least significant bit position. Assume that the exchange operation corresponds to replacing the data elements e_i and e_j present in PE[i] and PE[j] both by the product $e_i * e_j$. This can be done in O(1) time since PE[i] and PE[j] both are connected by the exchange links.

Let us assume the data elements e_i to be represented as the elements of a one dimensional array of length **n**. A sequence of **m** shuffle operations results in the pairs of data elements situated succesively $2^{(k-1)}$, $2^{(k-2)}$,..., 2^1, $2^0=1$ locations apart, being brought to adjacent PEs (proposition 1). After every exchange operation the data elements of any two adjacent PEs (connected by the exchange links) are replaced by the semigroup product of the data elements of the two PEs. If the shuffle and the exchange operations are performed alternately, such that each operation is performed **m** times, the data elements in all the PEs are replaced by the semigroup product over all the elements e_i in **O(m)** time.

Proposition 3 If 2^m PEs are connected as a shuffle-exchange network, any PE can communicate **B** data elements to all PEs in **O(B+m)** time.

Proof Fig. 2 illustrates the procedure to communicate data from PE[0] to all PEs. Every communication link is labelled with the number of the time step in which the data is communicated along the link. It can be seen that every link is used only once during the communication. Hence if B data elements are to be communicated from PE[i] to all PEs, the communication of the different data elements can be pipelined. In general data can be communicated from any PE[i] to all other PEs in time O(B+m).

B. Performing GJE on Shuffle-exchange

Mapping of the Matrix : As stated above the matrix **A** is assumed to be of size **nxn**. We assume without loss of generality that $n=2^k$. Let **r** denote the row index and **c** denote the column index of the elements of matrix **A**. Let the bit representation of **r** be $r_{(k-1)} r_{(k-2)} \dots r_1 r_0$ and the bit representation of **c** be $c_{(k-1)} c_{(k-2)} \dots c_1 c_0$. Assume that there are 2^{2p} PEs in the shuffle exchange network. The address of any PE is represented using **2p** bits. The element **A[r,c]** of the matrix is mapped onto the $PE[r_{(k-1)} r_{(k-2)} \dots r_{(k-p)} c_{(k-1)} c_{(k-2)} \dots c_{(k-p)}]$. Thus a partition of size $2^{k-p} \times 2^{k-p}$ is allocated to every PE.

Inputting the Matrix : Let bit representation of the address of a PE be denoted by $a_{(2p-1)} a_{(2p-2)} \dots a_p a_{(p-1)} \dots a_1 a_0$. Assume the I/O ports to be present in all the PEs with address denoted by $00 \dots 0 a_{(p-1)} \dots a_1 a_0$, such that the **p** higher order bits of the address are all zero. Thus for an **N** processor configuration , **N** processors are assumed to have I/O ports. The number of ports grows as the square root of the number of processors which is regarded as a practical assumption [4].

Every PE with address denoted by $00\ldots0a_{(p-1)}\ldots a_1a_0$ is used to input the elements of 2^{k-p} columns of the matrix A. The $n.2^{k-p}$ elements input to $PE[00\ldots0k_{(p-1)}\ldots k_1k_0]$ are sent to all the respective PEs using the COMMUNICATE-ROW operation described below. Using the COMMUNICATE-ROW operation it can be seen that the time required to input the matrix is $O(n.p.2^{k-p})$.

Time Required for ELIMINATE-COLUMN operation

1) COMMUNICATE-PIVOT(q) : The COMMUNICATE-PIVOT operation takes **$O(m)$** time by proposition 3.

2) FIND-PIVOT(q) : Every PE contains a submatrix of size $2^{k-p} \times 2^{k-p}$. Finding the maximum of the elements of the submatrix within the PE requires $O(2^{2(k-p)})$ time. Finding the maximum of all the elements $A[i,j]$, $i \geq q$, $j \geq q$, takes **$O(p)$** time by proposition 2. Thus the overall time required for the FIND-PIVOT step is $O(2^{2(k-p)}+p)$.

Before explaining the COMMUNICATE-ROW operation we assume that the bit representation of any p-bit integer r as $r_{p-1}\ldots r_1r_0$. Similarly the bit representation of other such variables as l, m, a is given by $l_{p-1}\ldots l_1l_0$, $m_{p-1}\ldots m_1m_0$ and $a_{p-1}\ldots a_1a_0$ respectively.

3) COMMUNICATE-ROW(q) : Let m be a number such that $m = q \text{ div } (2^{(k-p)})$ where div represents an integer division. Let a be a an integer $0 \leq a < 2^p$. The elements of the ROW[q] are present in the set of PEs denoted by $PE[m_{p-1}\ldots m_1m_0a_{p-1}\ldots a_1a_0]$ for all values of $a_{p-1}\ldots a_1a_0 = a$ such that $0 \leq a < 2^p$ (here a is just used as a running variable. In fact the above set of PEs contains the row of partitions denoted by PR[m] and hence this set of PEs is PE-R[m].

Consider any constant value of a = l such that the bit representation of l is $l_{p-1}\ldots l_1l_0$. To perform the COMMUNICATE-ROW operation we have to communicate the 2^{k-p} elements of the ROW[q] present in a $PE[m_{p-1}\ldots m_1m_0l_{p-1}\ldots l_1l_0]$ (belonging to the set PE-R[m]) to all the PEs belonging to PE-C[l]. The set of PEs denoted by PE-C[l] includes all PEs with address denoted by $PE[r_{p-1}\ldots r_1r_0l_{p-1}\ldots l_1l_0]$ for all values of $r_{p-1}\ldots r_1r_0 = r$ such that $0 \leq r < 2^p$.

Thus if q is the column to be communicated, $m = q \text{ div } 2^{k-p}$, l represents a p-bit constant value and r and a represent running variables such that $0 \leq r,a < 2^p$, then COMMUNICATE-COL(q) operation can be expressed as follows :

Assume that there are B data elements in every PE denoted by $PE[m_{(p-1)}\ldots m_1a_0{}_{(p-1)}\ldots a_1a_0]$, for a constant value m and for all values of a, $0 \leq a\ 2^p$ (the data in different PEs are different and in the above case $B = 2^{k-p}$). For performing COMMUNICATE-ROW operation the data elements of any $PE[m_{p-1}\ldots m_1m_0l_{p-1}\ldots l_1l_0]$ (for constant values m and l) are communicated to all PEs denoted by $PE[r_{p-1}\ldots r_1r_0l_{p-1}\ldots l_1l_0]$ for all values of r, $0 \leq r < 2^p$. We show below a method to perform the above computation in $O(Bp)$ time for a shuffle-exchange network of size 2^{2p}.

Let us first consider the case when B=1, i.e. when there is a single data element in all the PEs with address denoted by $m_{(p-1)}\ldots m_1m_0a_{(p-1)}\ldots a_1a_0$. Fig. 3 illustrates the required communication for a 16 processor shuffle-exchange network. As required in this proposition the data element present in the $PE[001_1l_0]$ is communicated to all the PEs with address denoted by $r_3r_2l_1l_0$, i.e. the PEs $00l_1l_0$, $01l_1l_0$, $10l_1l_0$ and $11l_1l_0$. In general, if we have a shuffle-exchange network of $n=2^{2p}$ PEs and there is a single data element in all the PEs with address denoted by $m_{(p-1)}\ldots m_1m_0a_{(p-1)}\ldots a_1a_0$ (the data element in different PEs being different) the required communication can be carried out by performing the following operations.

1) Perform shuffle and exchange operations alternately on the data elements such that each operation is performed p times. After this step, the data of $PE[m_{(p-1)}\ldots m_1m_0l_{(p-1)}\ldots l_1l_0]$ reaches all the PEs with address denoted by $l_{(p-1)}\ldots l_1l_0r_{(p-1)}\ldots r_1r_0$ for all values or r, $0 \leq r < 2^p$.

2) Perform the shuffle operation **p** times on the data present in all the PEs with address denoted by $l_{(p-1)}\ldots l_1l_0r_{(p-1)}\ldots r_1r_0$. After this step the data element of every PE, denoted by $l_{(p-1)}\ldots l_1l_0r_{(p-1)}\ldots r_1r_0$ reaches the PE, denoted by $r_{(p-1)}\ldots r_1r_0l_{(p-1)}\ldots l_1l_0$. The overall time required for the steps 1 and 2 is **$O(p)$**. Note that in the above communication operation the same links are being reused in the various steps (fig. 3). Hence if the above communication is to be performed when every PE with address $m_{(p-1)}\ldots m_1a_0{}_{(p-1)}\ldots a_1a_0$ has B data elements, all the communication steps have to be repeated for every data element. Hence the communication required in the proposition takes **$O(Bp)$** time.

Thus the COMMUNICATE-ROW(q) operation takes $O(2^{k-p}p)$ time.

4) COMMUNICATE-COL(q) : The COMMUNICATE-COL operation is similar to the COMMUNICATE-ROW operation. The time required for the COMMUNICATE-COL operation is $O(2^{k-p}.p)$.

5) UPDATE : The time required to update the $2^{k-p} \times 2^{k-p}$ elements of the matrix A present in every PE is

$O(2^{2(k-p)})$.

Thus the time required for the ELIMINATE-COLUMN operation is $O(2^{2(k-p)}+2^{(k-p)}.p+p)$.

Overall Time for Performing GJE The time required to perform ELIMINATE-COLUMN operation on all the n columns of the matrix is $O(n(2^{2(k-p)}+2^{(k-p)}.p+p))$. The value of **p** required to minimize the overall time is **p=logn-loglogn+w**. where **w** is a constant. For the value of **p** given by **p=logn-loglogn+w**, the time required to eliminate all **n** columns is $O(n\log^2 n)$. The time required to input the matrix is also $O(n\log^2 n)$.

Thus depending on the value of p expressed as a function of n which denotes the size of matrix, the following performances can be obtained on a shuffle-exchange network :

1) Optimal performance : GJE can be performed with asymptotic optimality (such that the product of (number of PEs)x(time required for the parallel implementation of GJE) is equal to $O(n^3)$) for values of p = k-logk+w = logn-loglogn+w, where w is a constant. For the above value of p, GJE can be performed in $O(n\log^2 n)$ time using $n^2/\log^2 n$ PEs.

2) Non-optimal performance : For values of p = k+w = logn+w, where w is a constant, GJE can be performed in O(nlogn) time using n^2 PEs.

The characteristics of the shuffle-exchange network used to perform GJE on a matrix of size nxn are given in Table I. The above mapping of the GJE on the shuffle-exchange network, does not attain the Ω(nlogn) time required (lower bound) to perform GJE in parallel. In the next section we present a modified shuffle-exchange interconnection scheme using which GJE can be performed in O(nlogn) time using n^2/logn processors.

5. MODIFIED SHUFFLE-EXCHANGE NETWORK

Assume that the total number of PEs to be interconnected is 2^{2m}. Every PE is represented as PE[i] where the bit representation of i is $a_{(2m-1)}\cdots a_{(m+1)}a_m a_{(m-1)}\cdots a_1 a_0$. Thus the bit representation of the address of any PE can be divided into two halves : the left half and the right half. The modified shuffle-exchange interconnection scheme can be considered as a combination of four interconnection schemes : shuffle-left, shuffle-right, exchange-left and exchange-right interconnection schemes, as defined below.

a) Shuffle-left Interconnection : In the shuffle-left interconnection scheme, the PE[i] is connected to the PE whose address is obtained by rotating the left half of the bit representation of i by one bit. Thus the PE, $a_{(2m-1)}\cdots a_m a_{(m-1)}\cdots a_1 a_0$ is connected by a one way link to the PE, $a_{(2m-2)}\cdots a_m a_{(2m-1)} a_{(m-1)}\cdots a_1 a_0$.

b) Shuffle-right interconnection : In the shuffle-right interconnection scheme, the PE[i] is connected to the PE whose address is obtained by rotating the right half of the bit representation of i by one bit. Thus the PE[$a_{(2m-1)}\cdots a_m a_{(m-1)}\cdots a_1 a_0$] is connected by a one way link to the PE[$a_{(2m-1)}\cdots a_m a_{(m-2)}\cdots a_1 a_0 a_{(m-1)}$].

c) Exchange-left Interconnection : In the exchange-left interconnection scheme the PE[$a_{(2m-1)}\cdots a_m a_{(m-1)}\cdots a_1 a_0$] is connected by a one way link to PE[$a_{(2m-1)}\cdots a'_m a_{(m-1)}\cdots a_1 a_0$] where a'_m is the binary complement of a_m.

d) Exchange-right Interconnection : In the exchange-right interconnection scheme the PE[$a_{(2m-1)}\cdots a_m a_{(m-1)}\cdots a_1 a_0$ is connected by a one way link to the PE[$a_{(2m-1)}\cdots a_m a_{(m-1)}\cdots a_1 a'_0$.

Thus it can be seen (fig. 4) that in the modified shuffle-exchange interconnection scheme, the 2^{2m} PEs are arranged as a 2-D array of 2^m x 2^m PEs where every row and every column of PEs is connected as a shuffle-exchange network. Thus every PE can be addressed as PE[i,j] where i (j) refers to the left (right) half of the bit representation of the address of the PE. Also, i and j represent the row number and column number respectively of the PE in the array of PEs shown in fig.4 Every PE is connected to atmost six other PEs using eight unidirectional links.

A. Properties of the Modified Shuffle-Exchange Network

Proposition 4 Let 2^{2p} PEs be connected as a modified shuffle-exchange network. Any PE can communicate B data elements to all PEs in O(B+p) time.

Proof Assume that B data elements are present in the PE[m,l] where m and l are integers $0 \leq m,l < 2^p$. the data elements can be broadcast to the mth row of PEs denoted by PE-R[m] in time O(B+p) by proposition 3. Then very PE of PE-R[m] can communicate in parallel its B data elements to all the PEs of its column. This requires O(B+p) time since evry column is connected as an independent shuffle-exchange network. Thus the overall communication takes $O(B+m)$ time.

Proposition 5 Let $n=2^{2p}$ PEs be connected as a modified shuffle-exchange network. Let the data element e_i be present in the PE[i], $0 \le i < 2^{2p}$. Any semigroup computation can be performed on all the elements e_i in time $O(p)$.

Proof (similar to the proof of proposition 2).

B. GJE on Modified Shuffle-Exchange Network

Mapping of the Matrix The mapping of the matrix A onto the modified shuffle-exchange network is the same as the mapping of matrix A onto the shuffle-exchange network presented in section 4. Thus the shuffle-exchange network is assumed to consist of 2^{2p} PEs and the address of any PE is represented using $2p$ bits.

Inputting the Matrix Let i be a number with bit representation of the form $a_{(2p-1)}a_{(2p-2)}\ldots a_{p}a_{(p-1)}\ldots a_1a_0$. Assume the I/O ports to be present in all the PEs with address denoted by $00\ldots0a_{(p-1)}\ldots a_1a_0$, such that the p higher order bits of the address are all zero. Every PE with address $00\ldots0a_{(p-1)}\ldots a_1a_0$ is used to input the elements of 2^{k-p} columns of the matrix A. The $n.2^{k-p}$ elements input to the PE$[00\ldots0k_{(p-1)}\ldots k_1k_0]$ are sent to the respective PEs as explained in the COMMUNICATE-ROW

operation. This communication takes $O(n(2^{k-p}+p))$ time as explained below. Thus the time required to input the matrix is $O(n(2^{k-p}+p))$.

Time Required for ELIMINATE-COLUMN operation

1) COMMUNICATE-PIVOT(q) : The COMMUNICATE-PIVOT operation takes $O(m)$ time by proposition 4.
2) FIND-PIVOT(q) : The time required for the FIND-PIVOT step is $O(2^{2(k-p)}+p)$ by proposition 5.
3) COMMUNICATE-ROW(q) : Let m be a number such that $m=q/(2^{(k-p)})$. All the elements of ROW[q] are present in the mth row of PEs in the array shown in fig.4. The mth row of PEs contains the mth row of partitions PR[m] and hence can be denoted by PE-R[m]. Every PE of PE-R[m] contains 2^{k-p} elements of ROW[q]. Also every PE-C is connected as an independent shuffle-exchange network. Thus every PE of PE-R[m] can communicate the elements of ROW[q] to all the PEs belonging to its column in time $O(2^{k-p}+p)$ by proposition 3.
4) COMMUNICATE-COL(k) : The COMMUNICATE-COL operation is similar to the COMMUNICATE-ROW operation and takes $O(2^{k-p}+p)$ time.
5) UPDATE : The time required to update the $2^{k-p}x2^{k-p}$ elements of the matrix A present in every PE is $O(2^{2(k-p)})$.

Thus the time required for the ELIMINATE-COLUMN operation is $O(2^{2(k-p)}+2^{(k-p)}+p) = O(2^{2(k-p)}+p)$.

Overall Time for Performing GJE The overall time required to perform ELIMINATE-COLUMN operation on all the n columns of the matrix is $O(n(2^{2(k-p)}+p))$. It can be verified that the value of p required to minimize the overall time is $p=\log n-(\log\log n)/2+w$. where w is a constant. For the value of p given by $p=\log n-(\log\log n)/2+w$, the time required to eliminate all n columns is $O(n\log n)$. The time required to input the matrix is $O(n\log n)$.

The characteristics of the modified shuffle-exchange network used for values of p given by $p=\log n-(\log\log n)/2+w$ are given in Table I.

Thus the performance of GJE on the modified shuffle-exchange network, of 2^{2p} PEs is asymptotically optimal for $0 < p < \log n-(\log\log n)/2$.

C. LAYOUT AREA FOR MODIFIED SHUFFLE-EXCHANGE

We first calculate the layout area assuming the area per PE to be O(1). In [7], Thompson proved that if N processors, connected using the shuffle-exchange interconnection are arranged in a straight line, the area occupied by the layout is $O(N^2/\sqrt{\log N})$. Such a layout of an N processor shuffle-exchange is a rectangle of length $O(N)$ and width $O(N/\sqrt{\log N})$. In the modified shuffle-exchange interconnection scheme, N processors are arranged as an array of $\sqrt{N} \times \sqrt{N}$ processors, where every row and every column of the array is connected as a shuffle-exchange network. The layout of a $\sqrt{N}$ processor shuffle-exchange is a rectangle of length $O(\sqrt{N})$ and width $O(\sqrt{N}/\log N)$. Thus if N PEs connected as the modified shuffle-exchange network are arranged as shown in fig. 4, the area required for the layout is $O(\sqrt{N}.\sqrt{N}/\log N) \times O(\sqrt{N}.\sqrt{N}/\log N) = O(N^2/\log N)$.

It can be seen that even if the area per PE is as large as $O(\sqrt{N}/\log N) \times O(\sqrt{N}/\log N)$, the layout area of the modified shuffle-exchange network remains $O(N^2/\log N)$.

6. CONCLUSIONS

It can be seen that the optimal performance of Gauss-Jordan elimination on the shuffle-exchange network requires $O(n\log^2 n)$ time using $n^2/\log^2 n$ PEs. The mapping of GJE which requires O(nlogn) time is also shown but it requires $O(n^2)$ PEs and hence is not optimal. It is not known if the GJE or GE algorithms can be mapped onto the shuffle-exchange network in O(nlogn) time with an optimal performance. But it can be seen that GJE can be performed in O(nlogn) time with an optimal performance on cube-connected cycles is known [7]. Thus as far as the best known performances of Gaussian or Gauss-Jordan elimination algorithms are concerned the performance of cube-connected cycles is better than that of shuffle-exchange network. Preparata [8] has shown that for the algorithms which can be classified under either ASCEND or DESCEND class, or for problems which can be performed using ASCEND or DESCEND procedures, the performance of CCC is same as that of shuffle-exchange. The Gaussian and Gauss-Jordan elimination algorithms cannot be performed using the ASCEND or DESCEND procedures and hence provide us a measure to compare the two architectures for problems with communication requirements entirely different from those required for ASCEND or DESCEND category of problems.

It is also shown that if the connectivity of the shuffle- exchange network is increased as in the modified shuffle exchange network, the GJE algorithm can be performed optimally in O(nlogn) time. As part of further research work, the suitability of the modified shuffle-exchange network proposed in this paper for applications other than solution of linear equations is being examined.

TABLE I

Item	Complexity for optimal performance shuffle-exchange	Complexity for non-optimal performance shuffle-exchange	Complexity for modified shuffle-exchange
Number of PEs used	$n^2/\log^2 n$	n^2	$n^2/\log n$
Memory/PE	$O(\log^2 n)$	O(1)	O(logn)
Number of PEs assumed to have I/O ports	n/logn	n	O(n/logn)
I/O time	O(nlogn)	O(n)	O(nlogn)
Overall time required to perform GJE	$O(n\log^2 n)$	O(nlogn)	O(nlogn)

REFERENCES

[1] G. H. Golub, and C. F. Van Loan, Matrix Computations, The Johns Hopkins Univ. press, 1985.
[2] P. R. Cappello, "A Mesh Automaton for Solving Dense Linear Systems," Proc. Intl. Conf. on Parallel Processing, Aug. 1985, pp. 79-86.
[3] Y. Saad, "Gaussian Elimination on Hypercubes," YALEU/DCS/RR-462, Mar. 1986.
[4] P. R. Cappello, "Gaussian Elimination on a Hypercube Automaton", Jl. of Parallel and Distributed Computing, 4, June 1987, pp. 288-308.
[5] H. S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE Trans. on Comput., C-20, 2, Feb. 71.
[6] S. G. Akl, Parallel Sorting Algorithms, Academic Press Inc., 1985.
[7] C. D. Thompson, A Complexity Theory for VLSI, Ph.D. Thesis, Dept. of Computer Sci., Carnegie Mellon Univ., 1980.
[8] K. Hwang, and F. A. Briggs, Computer Architecture and Parallel Processing, McGraw Hill Inc., 1984.
[9] H. T. Kung, "A Listing of Systolic Papers," Dept. of Computer Science, Carnegie Mellon Univ., Pittsburgh, Apr. 1986.
[10] C. Mead and L. Conway, Introduction to VLSI systems, Addison Wesley, Mass., 1980.
[11] R. B. Panwar, S. Raman and L. M. Patnaik, "Solution of Linear Equations on Cube-Connected Cycles," Proc. of the Third Intl. Conf. on Supercomputing, 1988.
[12] J. P. Preparata and J. E. Vuillemin, "The Cube-Connected Cycles : A Versatile Network for Parallel Computation," Comm. of the ACM, vol. 24, no. 5, May 1980.

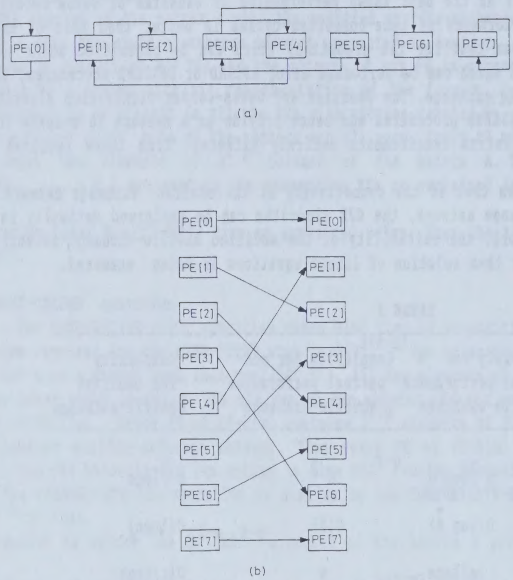

PE [0]
PE [1]
PE [2]
PE [3]
PE [4]
PE [5]
PE [6]
PE [7]
(a)
(b)

FIG. 1 PERFECT SHUFFLE INTERCONNECTION
(a) Interconnection of the PEs
(b) Interconnection shown as a mapping

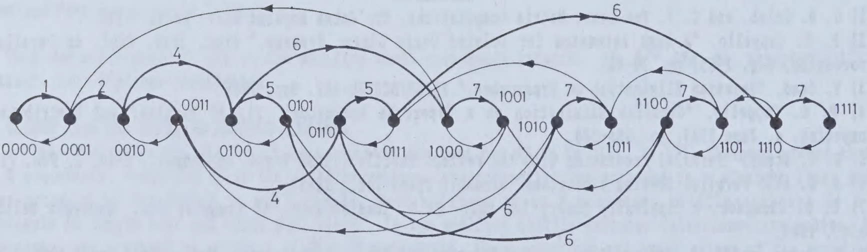

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
1
2
4
5
6
7

FIG. 2 COMMUNICATION PATTERN IN SHUFFLE – EXCHANGE NETWORK
(as required is proposition 2)

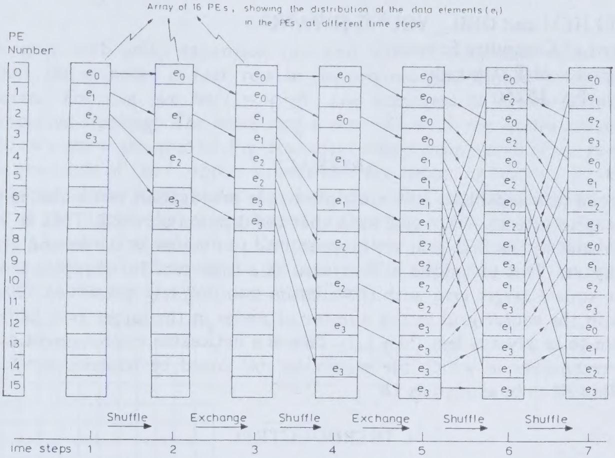

Array of 16 PEs, showing the distribution of the data elements (e_i) in the PEs, at different time steps
PE Number
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Shuffle
Exchange
Shuffle
Exchange
Shuffle
Shuffle
ime steps
1
2
3
4
5
6
7

FIG.3 COMMUNICATION PATTERN IN SHUFFLE – EXCHANGE NETWORK

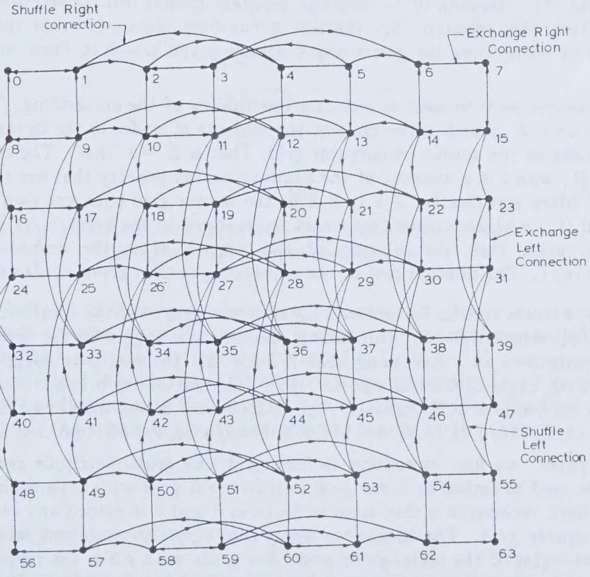

Shuffle Right connection
Exchange Right Connection
Exchange Left Connection
Shuffle Left Connection
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

FIG.4 THE MODIFIED SHUFFLE – EXCHANGE INTERCONNECTION SCHEME
(a 64 Processor configuration)

EMBEDDING RECTANGULAR GRIDS INTO SQUARE GRIDS
WITH DILATION TWO

RAMI MELHEM and GHIL_YOUNG HWANG
Department of Computer Science
The University of Pittsburgh
Pittsburgh, PA. 15260

ABSTRACT

In this paper, a new technique, the multiple ripple propagation technique, is presented for mapping an $h \times w$ grid into an $w \times h$ grid such that the dilation cost is 2. That is, such that any two neighboring nodes in the first grid are mapped into two nodes in the second grid that are at most distance 2 apart. This technique is then used as a basic tool for mapping any rectangular source grid into a square target grid with the dilation two property preserved. The ratio of the number of nodes in the source grid to the number of nodes in the target grid, called the expansion cost, is shown to be always less than 1.2. This is a noticeable improvement over the previously suggested techniques in which the expansion cost could be bounded by 1.2 only if the dilation cost is allowed to be as high as 18.

1. INTRODUCTION

In this research, we study the problem of squaring up a rectangular grid. That is, embedding an $h \times w$ rectangular grids into an $k \times k$ square grid, where $k \geqslant \lceil \sqrt{h\ w}\ \rceil$, and $\lceil\ \rceil$ is the ceiling function. The results of this research may be applied to the VLSI design of highly eccentric circuits that, without squaring up, would have to be laid out in a rectangular area with a height/width ratio very far from unity [6, 10]. They may also be applied to the mapping of rectangular problem domains (as for example finite element grids [3, 8]) into mesh connected architectures [1, 9]. Mapping of rectangular program graphs into hypercube architectures may also benefit from this research. Specifically, it has been shown [5] that this mapping may be accomplished by embedding the graph into a square graph which is, then, mapped easily to the hypercube.

Two measures may be used to estimate the quality of the embedding. The first measure is the expansion cost, E, which is the ratio of the number of nodes in the square target grid to the number of nodes in the source rectangular grid. That is $E = k^2/hw$. The other measure is the dilation cost D, which is a measure of the communication penalty that has to be paid due to the squaring up. More specifically, if a link λ in the source grid connects two neighboring nodes, say (i, j) and $(i, j+1)$, and these two nodes are mapped to the nodes (i', j') and $(i'+c_i, j'+c_j)$ in the target grid, then the dilation of the edge λ after the embedding is defined by $D(\lambda) = |c_i| + |c_j|$. The dilation cost of the embedding is then given by $D = \max\limits_{\lambda} D(\lambda)$.

The best known results for embedding an $h \times w$ grid into the smallest possible $k \times k$ grid are given in [2], where different embedding methods are suggested for different ranges of the eccentricity ratio $\rho = w/h$. Assuming that $h \geqslant 25$, all the methods suggested in [2] produce embeddings with expansion costs smaller than 1.2, and dilation costs ranging from 2 to 18, depending on the value of ρ. Specifically, the dilation cost is less than or equal to 3 if ρ is in one of the ranges $(1,2]$, $(10/3,4]$, $(8,9]$, or $(155,\infty)$. Otherwise, the dilation cost is larger than 5.

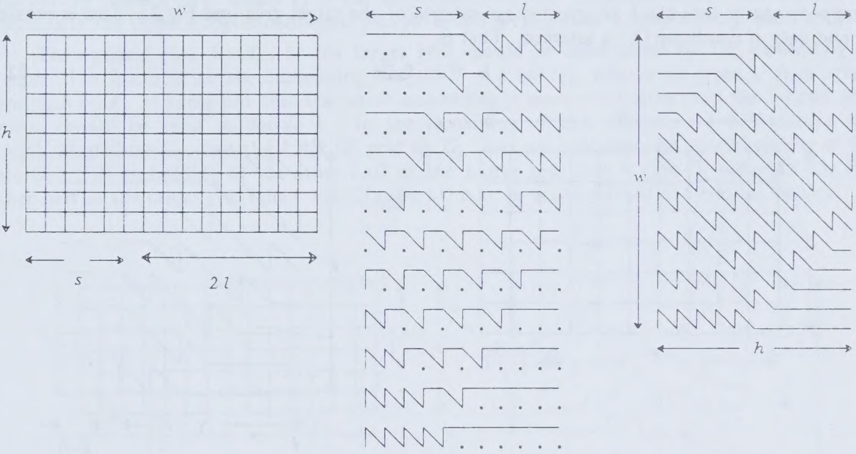
In this paper, we first introduce, in Section 2, the multiple ripple propagation technique which may be used to embed an $h \times w$ grid into an $w \times h$ grid with expansion cost 1 and dilation cost 2. This basic technique is then used in Sections 3 and 4 to embed any rectangular grid with $\rho \leqslant 4$ into a square grid. The idea is to apply the ripple propagation technique to carefully chosen subrectangles of the rectangular grid. For grids with $\rho > 4$, the ripple propagation technique may be combined with the well known technique of folding. This is described and analyzed in Section 5. Finally, in Section 6, we summarize our results and show that, it is always possible to square up any rectangular grid at a dilation cost of 2 and an expansion cost less than 1.2. This is a clear improvement over the results given in [2].

2. A MULTIPLE RIPPLE PROPAGATION TECHNIQUE

The purpose of the technique described in this section is to map an $h \times w$ grid, which satisfy

$$h < w \leqslant 2h \tag{1}$$

into an $w \times h$ grid with unity expansion cost and with dilation cost equal to 2. In order to accomplish that, the w nodes in each row in the original grid should be compressed to occupy only h columns. For this, we let $l = w - h$, and compress $2l$ nodes from each row into l columns by repeated rippling. The remaining $s = w - 2l = 2h - w$ nodes are left uncompressed. In Figure 1(b), we show the grid of Fig 1(a) after compressing each of its rows. As shown in the figure, the positions of the l ripples in each row are chosen as follows: In the first row, the l ripples are grouped to the right, and in the last row, the l ripples are grouped to the left. At each row, one of the ripples that was grouped to the right in the previous row, starts its propagation to the left (moves one column). The propagation of that ripple continues at a rate of one column every row until it can no longer propagate. The propagation of the ripples is very similar to the motion of the legs of a walking caterpillar.



(a) (b) (c)

Fig 1 - Embedding an 11×16 grid into an 16×11 grid
using multiple ripple propagation

Figure 1(b) is laid out to occupy $w + s$ rows and h columns. However, it may be noticed that s positions in each column is not utilized. This allows for the compression of Fig 1(b) into an $w \times h$ grid which has a dilation cost equal to 2 (see Fig 1(c)). In [7], a recursive formula is given for the function $F(i,j) = (u(i,j), v(i,j))$ which maps each point (i,j) in the source grid of Fig 1(a) to a corresponding point $(u(i,j), v(i,j))$ in the target grid of Fig 1(c). Moreover, it is proved that for any (i,j), where $i, j > 1$, the following is true

$$|u(i,j) - u(i-1,j)| + |v(i,j) - v(i-1,j)| \leqslant 2$$

and

$$|u(i,j) - u(i,j-1)| + |v(i,j) - v(i,j-1)| \leqslant 2$$

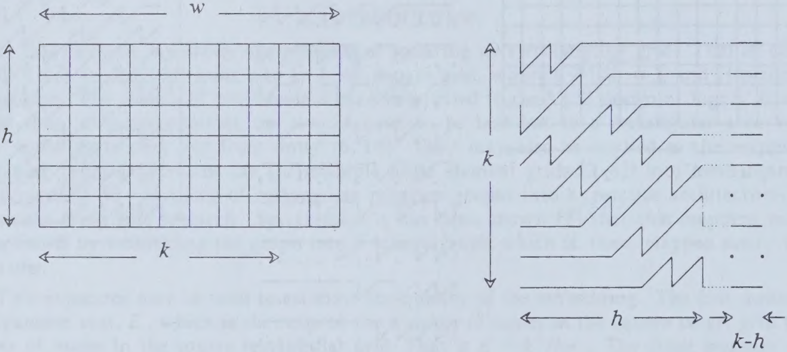
That is, any two adjacent nodes in the source grid are mapped into two nodes that are not more than a distance two apart in the target grid. This result proves that it is possible to map an $h \times w$ grid, $h < w \leqslant 2h$, exactly into an $w \times h$ grid with dilation cost 2. It is also possible to concatenate the $w \times h$ target grid with its symmetric image (reflected across the line $v = h$) to obtain an exact embedding of an $h \times 2w$ grid into an $w \times 2h$ grid with dilation cost 2. Along the same line of thinking, an $h \times 2w+1$ source grid may be divided into an $h \times w+1$ and an $h \times w$ subgrids. These two subgrids may then be embedded into an $w+1 \times h$ and an $w \times h$ grids, respectively, and by concatenating the former with the symmetric image of the latter, we may obtain an $w+1 \times 2h$ target grid. The dilation cost at the line of concatenation may be shown to be at most two. Grid concatenations of the type described here will be used repeatedly and tacitly in the rest of this paper.

In the following sections, we apply the above technique to our original problem of mapping an $h \times \rho h$ rectangular grid (ρ is assumed to be greater than unity), into a square grid. First, two basic methods are introduced for grids with $\rho \leqslant 4$. These methods are then combined with folding and applied effectively to the embedding of any grid with $\rho > 4$.

3. THE METHOD OF EXACT ROW FITTING

Let k, $k > h$ be the dimension of the square grid (called the target grid) onto which the given $h \times \rho h$ grid (called the source grid) is to be mapped. Of course, it is desirable to chose the smallest possible k in order to minimize the expansion cost $E = k^2/\rho h^2$. Given such a k, the method of exact row fitting assumes that the right most $h \times k$ subgrid of the source grid may be mapped exactly into the $k \times h$ right most subgrid of the target grid (see Fig 2). This is possible if and only if condition (1) is satisfied. That is

$$h \leqslant k \leqslant 2h \tag{2.a}$$



(a) (b)

Fig 2 - Embedding an 7×11 grid into an 9×9 grid using the method of exact row fitting

Moreover, if

$$k - h \geqslant \rho h - k \tag{2.b}$$

that is, the number of columns, $k-h$, remaining in the target grid is at least equal to the number of columns, $\rho h - k$, remaining in the source grid, then these columns may be mapped in a trivial way with dilation cost 2. In other words, the mapping may be completed with dilation cost 2 provided that the size of the target grid, k, satisfies the conditions (2.a/b).

The solution of the inequalities (2.a/b) may be found by, first, computing the minimum k that satisfies (2.b), and then checking that this value is consistent with (2.a). Specifically, (2.b) is satisfied if

$$k = K_r = \lceil \frac{\rho+1}{2} h \rceil \tag{3}$$

It is straight forward to check that the value of k given by (3) satisfies (2.a) if $\rho \leqslant 3 - 1/h$. Hence, the method of exact row fitting may be applied only if $\rho \leqslant 3 - 1/h$. Noting that $K_r \leqslant ((\rho+1)h+1)/2$, we may obtain an upper bound on the expansion cost of the resulting embedding. Namely,

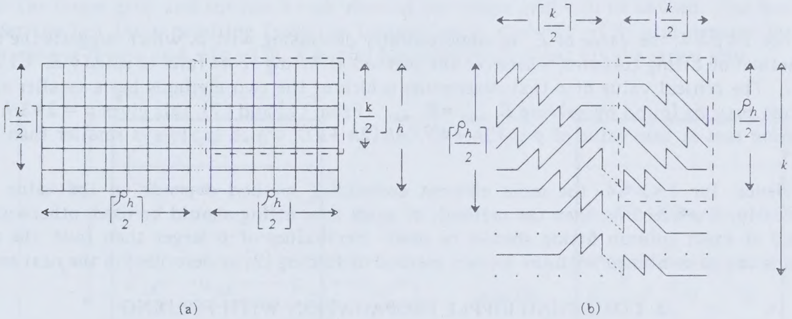
$$E_r = \frac{K_r^{\,2}}{\rho} \leqslant E_{r,max} = \frac{(\rho+1+\frac{1}{h})^2}{4\rho} \tag{4}$$

The value of E_r increases monotonically with ρ for $\rho > 1+1/h$, and hence may exceed 1.2 for large values of ρ. For example, assuming $h = 12$, then $E_r > 1.2$ if $\rho \geqslant 2.06$. Moreover, if $\rho > 3-1/h$, the method may not be applied. In these cases, the method of exact column fitting, described in the following section, can be used.

4. THE METHOD OF EXACT COLUMN FITTING

The embedding technique used in this section is based on the vertical disection of both the source and the target grids, each into two subgrids which are as equal as possible. Each of the source subgrids is then embedded into the corresponding target subgrid in a way that ensures that all the columns of the target grid are efficiently used. In order to deal with the case of ρh being an odd integer, the number of columns in the two source subgrids is taken to be $\lfloor \rho h/2 \rfloor$ and $\lfloor \rho h/2 \rfloor$, respectively, where $\lfloor\ \rfloor$ is the floor function. For the same reason, the number of columns in the target subgrid is divided into $\lfloor k/2 \rfloor$ and $\lfloor k/2 \rfloor$ columns, respectively (see Fig 3).

The optimal size, $k = K_c$, of the target grid should be determined by the embedding of $h \times \lfloor \rho h/2 \rfloor \rightarrow k \times \lfloor k/2 \rfloor$ or the embedding $h \times \lfloor \rho h/2 \rfloor \rightarrow k \times \lfloor k/2 \rfloor$, whichever gives a more strict condition on k. It turns out that the latter embedding is more restrictive than the former, and hence, should be used to derive k. In the remaining of this section, we will denote the $h \times \lfloor \rho h/2 \rfloor$ grid by G_s, and the $k \times \lfloor k/2 \rfloor$ grid by G_t, and we will describe an embedding of G_s into G_t. The embedding of the other half of the source grid (the $h \times \lfloor \rho h/2 \rfloor$ subgrid) into the other half of the target grid (the $k \times \lfloor k/2 \rfloor$ subgrid) may be accomplished in a similar fashion.



$\lceil \frac{k}{2} \rceil$
$\lfloor \frac{k}{2} \rfloor$
h
$\lceil \frac{\rho h}{2} \rceil$
$\lfloor \frac{\rho h}{2} \rfloor$
k
(a)
(b)

Fig 3 - Embedding an 7×15 grid into an 11×11 grid
using exact column fitting

Consider the upper $\lfloor k/2 \rfloor \times \lfloor \rho h/2 \rfloor$ subgrid of G_s, and embed it into the upper $\lfloor \rho h/2 \rfloor \times \lfloor k/2 \rfloor$ subgrid of G_t using the rippling propagation technique of Section 2. In order to accomplish this embedding we should have

$$\lfloor k/2 \rfloor \leqslant h \tag{5.a}$$

and condition (1) should be satisfied, namely

$$\lfloor k/2 \rfloor \leqslant \lfloor \rho h/2 \rfloor \leqslant 2\,\lfloor k/2 \rfloor \tag{5.b}$$

With this, each of the remaining $h - \lfloor k/2 \rfloor$ rows in G_s may then be compressed to have the same pattern as the last row of the $\lfloor \rho h/2 \rfloor \times \lfloor k/2 \rfloor \rightarrow \lfloor k/2 \rfloor \times \lfloor \rho h/2 \rfloor$ embedding. This results in a dilation cost equal to 2 and requires $2(h - \lfloor k/2 \rfloor)$ additional rows in G_t. Thus, the following should be satisfied

$$k - \lfloor \rho h/2 \rfloor \leqslant 2\,(h - \lfloor k/2 \rfloor) \tag{5.c}$$

Noting that $(x+1)/2 \geqslant \lfloor x/2 \rfloor \geqslant (x-1)/2$, we may calculate the minimum value of k which always satisfies (5.c). Namely,

$$k = K_c = \lceil \frac{\rho h/2 + 2h + 1}{2} \rceil \tag{6}$$

This value of K_c satisfies the conditions (5.a) and (5.b) as long as $\rho \leqslant 4$.

With the value of k given by (6), the two halves of the source grid may be successfully embedded into the two halves of the target grid with dilation cost 2. Noting that $\lfloor x/4 \rfloor \leqslant (x+3)/4$, it is possible to bound the expansion cost of the embedding as follows:

$$E_c \leqslant E_{c,max} = \frac{(\rho+4+\frac{5}{h})^2}{16\rho} \tag{7}$$

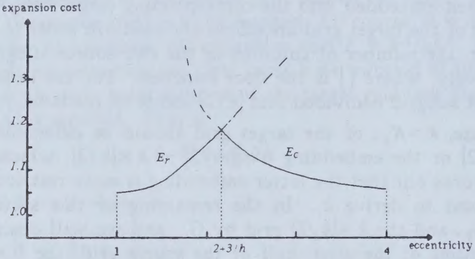


Fig 4 - Expansion cost Vs eccentricity for $1<\rho\leqslant 4$ $(h=20)$

For $1<\rho\leqslant 4$, the value of E_c is monotonically decreasing with ρ, which suggests the use of the method of fitting columns whenever the method of fitting rows fails to satisfy $E_r \leqslant 1.2$ (see Fig 4). The critical value of ρ that determines which of the two methods has a smaller expansion cost may be found by solving $E_{r,max} = E_{c,max}$. From (4) and (7), this gives $\rho = 2+3/h$. The expansion cost at this value of ρ is $(3h+4)^2/(4h(2h+3))$, which is always smaller than 1.2 if $h>17$.

Hence, for $1<\rho\leqslant 4$, the most efficient embedding method depends on the value of ρ. Specifically, if $\rho \leqslant 2+3/h$, then the methods of exact row fitting should be used, otherwise, the method of exact column fitting should be used. For values of ρ larger than four, the above methods can be combined with the known method of folding [2] as described in the next section.

5. COMBINING RIPPLE PROPAGATION WITH FOLDING

If $\rho = (q+1)^2$ for some integer $q \geqslant 1$, then the source grid may be folded $q+1$ times to fit exactly an $(q+1)h \times (q+1)h$ target grid. In fact, it is easy to show that if

$$\frac{(q+1)^2}{1.2} < \rho \leqslant (q+1)^2 \tag{8.a}$$

for some integer $q \geqslant 1$, then folding the source grid into an $(q+1)h \times (q+1)h$ target grid will result in an expansion cost less than 1.2. In Fig 5, we illustrate the technique of folding by an example. As clear from this figure, successive tracks (a track consists of h consecutive rows of the target grid) are joined by two $h \times h$ corner tiles that guarantee a dilation cost equal to two.

As described above, folding may result in few unused columns in the last track of the target grid, and condition (8.a) limits the number of these unused columns. It is also possible to apply folding and leave some rows of the target grid unused. More precisely, if the eccentricity of the source grid satisfies

$$q^2 \leqslant \rho \leqslant 1.2q^2 \tag{8.b}$$

then, it is possible to fold this grid into an $\rho h/q \times \rho h/q$ target grid. This will leave $(\rho-q^2)h/q$ unused rows in the target grid, and condition (8.b) will guarantee that the number of unused rows does not exceed $0.2qh$. Thus, the expansion cost will be less than 1.2.

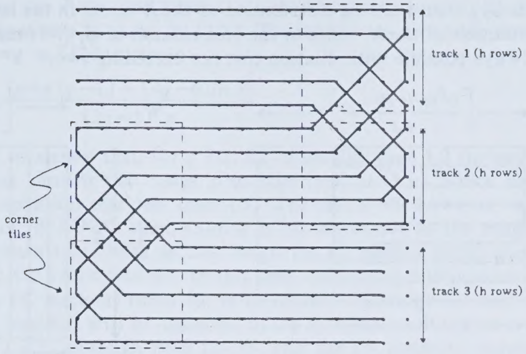
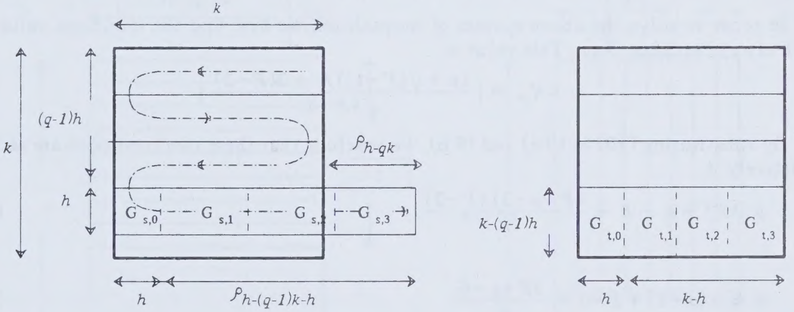


Fig 5 - Folding an 5×40 grid into an 15×15 grid

Combining folding with exact row fitting

Consider an $h \times \rho h$ source grid which satisfies $1.2q^2 < \rho < (q+1)^2/1.2$. Clearly, folding this grid into a square grid is too expensive (expansion cost larger than 1.2) because neither (8.a) nor (8.b) is satisfied. In this section, we introduce a method which combines folding and exact row fitting. This method will be denoted by FR. In order to describe the FR method we assume that the source grid is to be embedded into a target grid of size k, where k satisfies $qh \leqslant k \leqslant (q+1)h$. The embedding starts by folding the source grid into the target grid q times as shown in Fig 6(a). Clearly, the right most $h \times (\rho h - qk)$ subgrid of the source grid will not fit into the target grid, and the last $k - qh$ rows of the target grid will be unused. The idea is to consider the last track resulting from the folding (an $h \times \rho h - (q-1)k$ grid denoted by G_s), and to squeeze it into an $k - (q-1)h \times k$ grid (denoted by G_t) that fits the target grid.



(a) (b)

Fig 6 - Combining folding with exact row fitting ($q = 4$, and $P = 4$)

The squeeze is performed by partitioning G_s vertically into P subgrids $G_{s,0}, \cdots, G_{s,P-1}$, and partitioning G_t vertically into P subgrids, $G_{t,0}, \cdots, G_{t,P-1}$, and then mapping each $G_{s,i}$ into the corresponding $G_{t,i}$. The partitioning of G_s is such that $G_{s,0}$, is an $h \times h$ grid and each remaining $G_{s,i}$, $i = 1, \dots, P-1$ is an $h \times \lfloor (\rho h - (q-1)k - h)/(P-1) \rfloor$ grid. Note that if $\rho h - (q-1)k - h$ does not divide $P-1$ then $G_{s,P-1}$ will have few empty columns. Similarly, the partition of G_t is such that $G_{t,0}$ is an $k - (q-1)h \times h$ grid and each of the remaining $G_{t,i}$, $i = 1, \dots, P-1$ is an $k - (q-1)h \times \lfloor (k-h)/(P-1) \rfloor$ grid.

The method of exact row fitting introduced in Section 3 is used to map each $G_{s,i}$, $i = 1, \dots, q-1$, into the corresponding $G_{t,i}$. As for the mapping $G_{s,0} \rightarrow G_{t,0}$, it should ensure that the transition from track $q-1$ to track q does not increase the dilation cost beyond two. This may be accomplished by expanding the $h \times h$ corner tiles (see Fig 5) into an $k - (q-1)h \times h$

pattern that fits $G_{t,0}$ such that the distribution of the h nodes in the last column of $G_{t,0}$ is similar to the distribution of the h nodes in the first column of $G_{t,1}$. It may be shown that such an expansion is always possible with dilation cost not exceeding two.

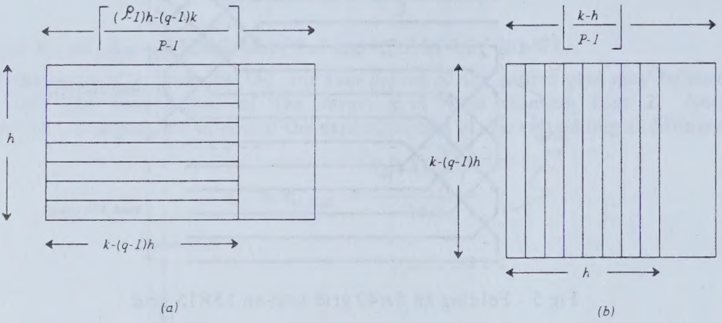


Fig 7 - Mapping $G_{s,i}$ into $G_{t,i}$ in the FR method

In order to compute the optimum size k of the target grid, we follow the same reasoning as in section 3. Specifically, the method of exact row fitting may be used for mapping any $G_{s,i}$ into the corresponding $G_{t,i}$. For this, the following conditions should be satisfied (see Fig 7):

$$h \leqslant \left\lfloor \frac{k-h}{P-1} \right\rfloor \tag{9.a}$$

$$h \leqslant k-(q-1)h \leqslant 2h \tag{9.b}$$

$$\left\lfloor \frac{k-h}{P-1} \right\rfloor - h \geqslant \left\lceil \frac{(\rho-1)h-(q-1)k}{P-1} \right\rceil - (k-(q-1)h) \tag{9.c}$$

In order to solve the above system of inequalities, we first find the minimum value of k which always satisfies (9.c). This value is:

$$k = K_{fr} = \left\lceil \frac{(\rho+q(P-1))\,h + 2(P-2)}{q+P-1} \right\rceil \tag{10}$$

By substituting (10) in (9.a) and (9.b), we conclude that these two conditions are satisfied, respectively if

$$\rho \geqslant P^2 + q - P + \frac{(P+q-3)\,(P-2)}{h} \tag{11.a}$$

and

$$\rho \leqslant q\,(q+1) + P-1 - \frac{3P+q-6}{h} \tag{11.b}$$

Hence, given any rectangular grid, the method may be used if there exists a P that satisfies (11). The number of partitions P also affects the expansion cost. More precisely, from (10), we find that $K_{fr} \leqslant ((\rho h + q(P-1)h + 3P + q - 6)/(q+P-1)$, which may be used to bound the expansion cost by

$$E_{fr} \leqslant E_{fr\,max} = \frac{(\rho + q(P-1) + (3P+q-6)/h)^2}{(P+q-1)^2\,\rho} \tag{12}$$

The derivative $\partial E_{fr\,max}/\partial P$ is negative for $\rho \geqslant q^2$ which means that, from the point of view of minimizing E_{fr}, it is advantageous to find the maximum P which satisfies (11). For $P > q$, the two conditions (11.a) and (11.b) may not be satisfied simultaneously. If, however, ρ satisfies

$$q^2 + \frac{(q-2)(2q-3)}{h} \leqslant \rho \leqslant (q+1)^2 - 2 - \frac{4q-6}{h}\,, \tag{13}$$

then, (11.a/b) are satisfied for $P=q$, and hence the embedding may be completed with q partitions in a target grid whose size is given by (10). The maximum expansion cost may then be found by substituting $P=q$ in (12) to obtain

$$E_{fr} \leqslant E_{fr,max} = \frac{(\rho+q(q-1)+(4q-6)/h)^2}{(2q-1)^2\,\rho} \tag{14}$$

In Section 6, it will be shown that, for $q \geqslant 3$, E_{fr} is smaller than 1.2 for any ρ in the range specified by (13), and that, outside that range, ρ satisfies (8.a) or (8.b), which means that folding may be used with expansion cost less than 1.2. The case $q=2$, however, is slightly more complicated. For instance, if $h \geqslant 20$, then, folding is too expensive in the range $5.85 \leqslant \rho \leqslant 7.5$ (expansion cost is larger than 1.2). Also, in that range, the FR method either does not apply (if $\rho > 6.9$) or gives $E_{fr} > 1.2$ (if $5.85 < \rho < 6.9$). In this case, combining folding with the method of exact column fitting (the FC method) turns out to be useful. Although we only need this combination for $q=2$, the FC method will be described in the next section for general q. The reason for doing so is that for $q \geqslant 3$, although both the FC and the FR methods realize an expansion cost less than 1.2, it will be shown that the FC method gives better results than the FR method for some subranges of ρ.

Combining folding with exact column fitting

In this method, denoted from now on by FC, the source grid is folded into the target grid as described in the previous section, and also each of G_s and G_t is partitioned into P subgrid. The FC method is different from the FR method in that each subgrid $G_{s,i}$, $i=1,...,P-1$, is mapped into the corresponding $G_{t,i}$ using the method of exact column fitting rather than exact row fitting.

The conditions that have to be satisfied in order to map $G_{s,i}$ into $G_{t,i}$ using exact column fitting are analogous to the conditions (5.a/b/c) of Section 4. Specifically, these conditions are (refer to Fig 8):

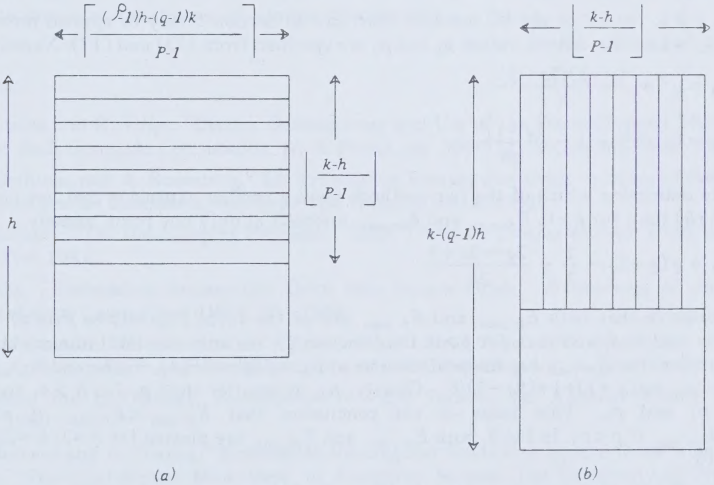


Fig 8 - Mapping $G_{s,i}$ into $G_{t,i}$ in the FC method

$$\left\lfloor \frac{(\rho-1)h-(q-1)k}{P-1} \right\rfloor \leqslant k-(q-1)h \tag{15.a}$$

$$\left\lfloor \frac{k-h}{P-1} \right\rfloor \leqslant \left\lfloor \frac{(\rho-1)h-(q-1)k}{P-1} \right\rfloor \leqslant 2\left\lfloor \frac{k-h}{P-1} \right\rfloor \tag{15.b}$$

$$k-(q-1)h - \left\lfloor \frac{(\rho-1)h-(q-1)k}{P-1} \right\rfloor \geqslant 2\,(h - \left\lfloor \frac{k-h}{P-1} \right\rfloor \tag{15.c}$$

The same technique that was used in the last sections is applied to the solution of the above inequalities. From (15.c), the minimum size of the target grid is found to be

$$k = K_{fc} = \left\lceil \frac{(\rho+qP+P-q)h + 3(P-2)}{P+q} \right\rceil \tag{16}$$

By substituting (16) in (15.a) we obtain the condition

$$\rho \leqslant P^2-P+q + (P+q-3)(P-2)/h$$

which may be satisfied only if $P>q$. Also, by using (16) to compute the expansion cost E_{fc}, and then differentiating the resulting formula, we find that $\partial E_{fc}/\partial P$ is positive for $\rho \leqslant P^2-P+q$. This means that using $P=q+1$ partitions will give the best expansion cost. Now using $P=q+1$ in (16), and substituting the result in (15.a) and (15.b), we find that these conditions are satisfied if ρ lies in the following range

$$q^2+1+\frac{3(q-1)}{h} \leqslant \rho \leqslant (q+1)^2-1-\frac{(q-1)^2}{qh} \tag{17}$$

That is, the FC method may be applied if ρ satisfies (17). The expansion cost may then be computed from (16) with $P=q+1$. The upper bound on this cost is given by

$$E_{fc} \leqslant E_{fc,max} = \frac{(\rho+q(q+1)+1+(5q-3)/h)^2}{(2q+1)^2\rho} \tag{18}$$

6. DISCUSSION AND CONCLUSION

Given an $h \times \rho h$ source grid, let q be the integer that satisfies $q^2 \leqslant \rho < (q+1)^2$. For $q=1$, it has been shown in Sections 3 and 4 that the mapping of the source grid into a square rectangular grid may be accomplished by using the method of exact row fitting if $\rho \leqslant 2+3/h$, or the method of exact column fitting if $\rho > 2+3/h$. In both cases, the expansion cost is proven to be less than 1.2.

For, $q \geqslant 2$, the FR or the FC methods described in Section 5 may be applied provided that $\rho_1 < \rho < \rho_2$, where the critical values ρ_1 and ρ_2 are specified from (13) and (17). Namely,

$$\rho_1 = q^2 + \frac{(q-2)(2q-3)}{h} \tag{19.a}$$

$$\rho_2 = (q+1)^2 - 1 - \frac{(q-1)^2}{qh} \tag{19.b}$$

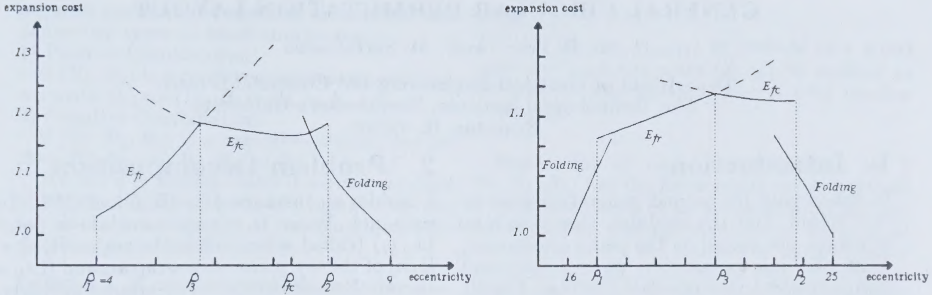
In order to determine which of the two methods gives a smaller expansion cost, we notice from (14) and (18) that, for $\rho > 0$, $E_{fr,max}$ and $E_{fc,max}$ intersect at only one point, namely

$$\rho_3 = q(q+1) - \frac{1}{2} + \frac{2q^2-3q+9}{2h} \tag{19.c}$$

We also observe that both $E_{fr,max}$ and $E_{fc,max}$ are of the form $f(\rho)=(\rho+a)^2/(b\rho)$, for some constants a and b. Given that, for $\rho>0$, the function f has only one local minima at $\rho=a$, we may determine that $E_{fr,max}$ has its local minima at $\rho_{fr}=q(q-1)+(4q-6)/h$, and $E_{fc,max}$ has its minima at $\rho_{fc}=q(q+1)+1+(5q-3)/h$. Clearly, ρ_{fr} is smaller than ρ_1 for $h>4$, and ρ_{fc} lies between ρ_1 and ρ_2. This leads to the conclusion that $E_{fc,max} < E_{fr,max}$ if $\rho > \rho_3$ and $E_{fc,max} > E_{fr,max}$ if $\rho < \rho_3$. In Fig 9, both $E_{fc,max}$ and $E_{fr,max}$ are plotted for $q=2$, $h=20$, and for $q=4$, $h=10$.

Hence, if ρ lies between ρ_1 and ρ_3, the FR method is recommended, and if ρ lies between ρ_3 and ρ_2, then the FC method is recommended. If this strategy is applied, then the largest expansion cost occurs at either $\rho=\rho_2$ or $\rho=\rho_3$. By direct substitution of (19.b) and (19.c) into (18), and after simple algebraic manipulation, it may be shown that, for $h \geqslant 18$, the value of $E_{fc,max}$ is less than 1.2 at ρ_2 and ρ_3.

Neither the FR method nor the FC method may be applied if ρ is less than ρ_1 or larger than ρ_2. However, in these two cases, the expansion cost resulting from simple folding is low because ρ is close enough to q^2 and $(q+1)^2$, respectively. In fact, if $\rho \leqslant \rho_1$, then $\rho < q^2(1+2/h)$, which satisfies (8.b) if $h>10$. Also, if $\rho \geqslant \rho_2$, then $\rho \geqslant (q+1)^2(1-1/qh)-1$, which satisfies (8.a)



(a) $q=2, h=20$ (b) $q=4, h=10$

Fig 9 - Expansion cost for $q^2 \leqslant \rho < (q+1)^2$

if $h > 10$. In other words, the application of simple folding in these two regions will result in an expansion cost less than 1.2.

In brief, new techniques have been presented and analyzed in this paper, for embedding an $h \times \rho h$ rectangular grid into a square grid with dilation cost equal to two. The most appropriate technique for a given grid have been shown to depend on the size of that grid, that is on h and ρ. By adhering to the selection strategy suggested in the paper, the expansion cost is guaranteed to be smaller than 1.2 if h is larger than or equal to 18.

Finally, we should mention that, recently, the same problem of embedding rectangular grids into square grids have been studied independently by Ellis [4]. Specifically, it is shown in [4] that it is possible, with dilation at most three, to embed any $h \times w$ grid into an $k+1 \times k+1$ grid, where $k = \lfloor \sqrt{hw} \rfloor$. This leads to an expansion cost which is smaller than the one resulting from our technique at the expense of a larger dilation cost.

References

1. L. Adams and R. Voigt, "Design, Developmnet and Use of the Finite Element Machine," in *Large Scale Scientific Computation*, ed. S. Parter, pp. 301-321, Accademic Press, 1984.
2. R. Aleliunas and A. Rosenberg, "On Embedding Rectangular Grids in Square Grids," *IEEE Transactions on Computers*, vol. C-31, no. 9, pp. 907-913, 1982.
3. S. Bokhary, "On the Mapping Problem," *IEEE Trans on Computers*, vol. C-30, no. 3, pp. 207-214, 1981.
4. J. Ellis, "Embedding Rectangular Grids into Square Grids," *Proceedings of the Aegean Workshop on Computing (AWOC 88)*, 1988.
5. S. L. Johnsson, "Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures," *J. of Parallel and Distributed Computing*, vol. 4, pp. 133-172, 1987.
6. C. Mead and L. Conway, *Introduction to VLSI Systems*, pp. Addison-Wesley, Reading, MA, 1980.
7. R. Melhem and G. Hwang, "Embedding Rectangular Grids into Square Grids with Dilation Two," Technical Report 88-4, Dept. of Computer Science, The University of Pittsburgh, March 1988.
8. P. Sadayappan and F. Ercal, "Nearest-Neighbor Mapping of Finite Element Graphs onto Processor Meshes," *IEEE Trans. on Computers*, vol. C-36, no. 12, pp. 1408-1424, 1987.
9. P. Tseng, K. Hwang, and P. Kumar, "A VLSI Based Multiprocessor Architecture for Implementing Parallel Algorithms," *Proc. of The International Conference on Parallel Processing*, 1985.
10. L. Valiant, "Universality Considerations in VLSI circuits," *IEEE Trans. on Computers*, vol. C-30, pp. 135-140, 1981.

GENERAL CIRCULAR PERMUTATION LAYOUT

R. D. Lou and M. Sarrafzadeh

Department of Electrical Engineering and Computer Science
The Technological Institute, Northwestern University
Evanston, IL 60208

1 Introduction

In sea-of-gate (or second generation gate array) layout, first the modules, viewed as a set of points, are placed on the plane (*placement*). Next the interconnection among the modules is realized (*intermodule routing*). Finally, *detailed routing* around the modules(*circular routing*) is obtained. In sea-of-gate layout each circular routing problem is completed over a module and thus it is a sigle-layer *circular permutation layout problem* (CPL), for there is only one (metal) layer over each module. CPL is used also in printed circuit board environment.

Formally, in CPL problem there are two concentric circles, C_{in} and C_{out}, with C_{in} being the inner circle (representing a module) and C_{out} being the outer circle. There are a set of n inner terminals on C_{in} and a set of n outer terminals on C_{out}. Given a permutation $\Pi = (\pi_1, \pi_2, ..., \pi_n)$, a *net* $N_i = (i, \pi_i)$ is a pair of inner terminal i and outer terminal π_i te be connected by means of a *wire*. A *window* $W(i)$ is the portion of C_{in} between inner terminals i and $i + 1 (mod\ n)$. If a wire does not pass through any window then it is called a *direct wire*; otherwise, it is a *indirect wire*. There are the following assumptions: $A1$) There is only one layer available for wiring. $A2$) Each wire can intersect C_{in} at most once (to avoid "long" wires). $A3$) At most K wires can pass through a window. $A4$) All wires cannot cross C_{out}. $A5$) Wires are pairwise nonintersecting.

An instance of CPL is specified by a permutation Π and an integer K : we denote it as $\eta = (\Pi, K)$. A solution to an arbitrary instance of CPL is a single-layer realization if one exists. Previously, only $\eta = (\Pi, 1)$ has been studied [O,RNMN]. An $O(n^2)$ algorithm was presented in [O], where n is the total number of nets. More recently, a linear time algorithm was proposed [RNMN]. In this article, we devise a linear time algorithm to solve an arbitrary instance of CPL with a fixed $K \geq 1$; That is why we call it "General" CPL problem. We proceed as follows: In section 2, we use an algorithm to decompose an instance into simpler problems (blocks). Then, in section 3, we lay out the blocks and combine the layout. With a trivial modification, we can extend the method to an arbitrary instance of CPL with different K for different window. For a complete manuscript see [LS].

2 Problem Decomposition

Consider an instance $\eta = (\Pi, K)$ of CPL. To each net $N_i = (i, \pi_i)$ we associate a point (x_i, y_i) (called *net-point*) in the region $[1, n] \times [1, n]$ of the x-y plane with wrap around (i.e., a torus). Because of the wrap around property, all additions and substractions are module n.

Generalizing the notion of point domination [PS], we say that points $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$ *dominate each other* if $x_i > x_j$ and $y_i > y_j$. If P_i and P_j do not dominate each other, then we say P_i and P_j *cross each other.* Consider a point sequence $\rho = (P_1, P_2, ..., P_m)$, where $P_i = (x_i, y_i)$. The point sequence ρ is called a *chain* if the points dominate each others. The chain ρ is a *continuous chain* if $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i + 1$ for $1 \leq i < m$. A continuous chain is a *fundamental chain*, if no other continuous chain contains it. The point sequence ρ is an *antichain* if the points cross each others. The definition of *fundamental antichain* is similar to the definition of fundamental chain.

A *block* $B = (l, r, b, t)$ is a rectangular region of the x-y plane, bound by two vertical lines $x = l$ (left) and $x = r$ (right), and two horizontal lines $y = b$ (bottom) and $y = t$ (top), where $l \leq r$ and $b \leq t$. The wires passing through $W(l - 1)$ (or $W(r)$) are called the *left side-wires* (or the *right side-wires*) of the block. Point $P_i = (x_i, y_i)$ and its corresponding wire are said to be "of the block B" if $l \leq x_i \leq r$ and $b \leq y_i \leq t$. A block, which contains all net-points of an instance is called an *universal block* of the instance. A *fundamental chain (or antichain) block* is a block with a fundamental chain (or antichain) in its diagonal.

Consider a block sequence $\beta_1 = \langle B_1, B_2 \rangle$, where B_1=(l_1, r_1, b_1, t_1) and B_2=(l_2, r_2, b_2, t_2). Block sequence β_1 is called a *positive-combination set* if $l_2 = r_1 + 1$ and $b_2 = t_1 + 1$. Block sequence $\beta_2 = \langle B_1, B_2, B_3 \rangle$, where B_1= (l_1, r_1, b_1, t_1), B_2= (l_2, r_2, b_2, t_2), and B_3= (l_3, r_3, b_3, t_3), is called a *negative-combination set* if $l_2 = r_1 + 1$, $t_2 = b_1 - 1$, $l_3 = r_2 + 1$, $t_3 = b_2 - 1$, and both B_1 and B_3 are fundamental antichain blocks. When B_3 (or B_1) does not fit the conditions stated above, for consistency, we introduce a fictitious block $B'_3 = (r_2, r_2, b_2, b_2)$ (or $B'_1 = (l_2, l_2, t_2, t_2)$), "without size", called *null* block. We say that $\langle B_1$, B_2, $B_3 \rangle$ (or $\langle B_1$, B_2, $B'_3 \rangle$ or $\langle B'_1$, B_2, $B_3 \rangle$), is a negative-combination set. B_1(or B'_1), B_2, and B_3(or B'_3) are called the *upper, middle,* and *lower* block of the negative-combination set, respectively. B_1(or B'_1) and B_3(or B'_3) are

---

[0] This work was supported in part by the National Science Foundation under Grant MIP-8709074.

negative combination mate of each other. We define two types of block combination:

1) Positive Combination:

if $\langle B_1, B_2\rangle$ is a positive-combination set then we write B_{12}=$+(B_1, B_2)=(l_2, r_1, b_2, t_1)$.

2) Negative Combination:

if $\langle B_1, B_2, B_3\rangle$ is a negative-combination set then B_{123}=$-(B_1, B_2, B_3) = (l_1, r_3, b_3, t_1)$.

We say η is *decomposable* if we can combine all the fundamental blocks of η into one universal block by a sequence of combinations; otherwise, η is nondecomposable.

Lemma 1 *A nondecomposable instance $\eta = (\Pi, K)$ of CPL does not have a solution.*

Because positive combination is an associative operation, $+(+(B_1, B_2), B_3)$ $=+(B_1,$ $+(B_2, B_3))$ $=+(B_1,$ $B_2,$ $B_3)$. Also, because of the wrap around property of the x-y plane region under consideration, if an instance is decomposed into a sequence of combinations of blocks, the sequence of combinations and blocks can be rotated. For example, consider a universal block of an instance B_0= $+(B_1,\ldots,$ $B_j,$ $-(B_{j+1},$ $B_{j+2},$ $B_{j+3}),$ $B_{j+4},\ldots,$ $B_k)$, then B_0= $+(B_k$,$B_1,\ldots,$ $B_j,$ $-(B_{j+1},$ $B_{j+2},$ $B_{j+3}),$ $B_{j+4},\ldots,$ $B_{k-1})$ =......= $+(B_{j+2},$ $-(B_{j+3},$ $+($ $B_{j+4},..,$ $B_k,$ $B_1,..,$ $B_j)$ $B_{j+1}))$. We represend these equivalent combinations with rotation property by B_0 =$[B_1,$ $B_2,..,$ $B_j,$ $B_{j+1}(B_{j+3}),$ $B_{j+2},$ B_{j+3} $(B_{j+1}),$ $B_{j+4},\ldots,$ $B_k]$, called *the combination equation* of the instance. In this equation, operators are omitted and negative-combination mate of each upper or lower block of negative-combination sets is denoted by the block name in ( ) following each block.

Lemma 2 *The combination equation of a decomposable instance of CPL is unique.*

We have algorithm COMBINATION (omitted here).

Theorem 1 *Algorithm COMBINATION finds the combination equation of an instance of CPL, if it exists, and runs in linear time.*

3 Layout Algorithm

In this section first we study the layout properties of blocks and block combinations.

Lemma 3 *The realization of a fundamental chain block is a set of direct wires.*

Lemma 4 *A fundamental antichain block in a positive combination set has at least the following two realizations[RNMN], where $\lceil x\rceil$ denotes ceiling of x.*

L1) Let $(x_{\lceil (m+1)/2\rceil}, y_{\lceil (m+1)/2\rceil})$ be realized as a direct wire and each net-point (x_i, y_i) be realized as an indirect wire passing through window $W(x_{m-i})$.

L2) Let $(x_{\lceil m/2\rceil}, y_{\lceil m/2\rceil})$ be realized as a direct wire and each net-point (x_i, y_i) be realized as an indirect wire passing through each window $W(x_{m-i})$.

Realization of negative-combination set $\langle B_1, B_2, B_3\rangle$ has the following three properties : $P1$) All net-points of B_1 are realized as indirect wires passing through windows of B_3, and all net-points of B_3 are realized as indirect wires passing through windows of B_1. $P2$) Either $W(l_1)$ or $W(r_3)$, but not both, has wires passing through. $P3$) Either $W(r_1)$ or $W(r_2)$, but not both, has wires passing through.

Lemma 5 *Any realization of a negative- combination set has properties $P1$, $P2$, and $P3$.*

According to $P2$ and $P3$, wires may or may not pass through windows $W(l_1)$, $W(r_1)$, $W(r_2)$, and $W(r_3)$. There are only four possible cases. In order to find realizations of a block with the minimum number of upper or lower side-wires among all possible realizations, we have to consider at most four cases. According to previous lemmas, we construct algorithm LAYOUT. Given a block B,may or may not combined from other blocks, algorithm LAYOUT finds realizations of B with the minimum number of upper or lower side-wires among all possible realizations of B, and takes time propotional to the number of the net-points of the block.

Theorem 2 *Given an instance of CPL, algorithm LAYOUT ,with algorithm COMBINATION, finds a realization of the instance, if one exists, and runs in linear time.*

References

[LS] Lou, R.D. and Sarrafzadeh, M., "General Circular Permutation Layout," manuscript, Department of Electrical Engineering and Computer Science, Northwestern University, 1988.

[O] Ozawa, T., "A Routing Procedure for an IC Module with Many Pins - A Solution to a Circular Permutation Layout Problem," *Network*, vol. 15, pp. 33-48. 1985.

[RNMN] Rim, C.S., Naclerio, N.J., Masuda, S. and, Nakajima, K., "A Linear Time Algorithm for Circular Permutation Layout," Allerton Conference, 1987.

[PS] Preparata, F.P. and Shamos, M.I., "Computational Geometry," Springer-Verlag, 1985.

THE ASSIGNMENT HEURISTIC FOR CROSSING REDUCTION IN BIPARTITE GRAPHS

TIZIANA CATARCI
Dipartimento di Informatica e Sistemistica - Universita` degli Studi di Roma "La Sapienza"
Via Buonarroti 12 - 00185 Roma Italy

INTRODUCTION

Many recent works try to develop models and methods for designing VLSI chips. Most of these efforts have been directed towards producing network layout which minimize the area consumed by the chip. Leighton [11] shows that the layout problem for an arbitrary graph can be reduced to that for a planar graph. Moreover, he proves that the number of crossings and the wire area of a graph are lower bounds and are also close to an upper bound on the layout area of any graph [13]. Hence, arc crossings and wire area arguments are very helpful in providing a good graph layout. The general problem of arc crossings minimization is NP-complete [7]. Also, no efficient heuristics are known for crossing minimization in general graphs. In this work we deal with crossings minimization for an important class of graphs, ***bipartite graphs***, tipically used in routing problems ([8] [1] [9]).

A bipartite graph (or *2-layered network*) is a directed graph whose vertex set is partitioned into two layers. Formally, a bipartite graph is a triple G = (L, R, E), where L and R are two disjoint sets of vertices, $|L| = n$ (the left part) and $|R| = m$ (the right part), and $E \subset = L \times R$ is the set of edges (or arcs). In the following, we assume that G is connected, i.e. we ignore isolated vertices.

We consider drawings of G such that: 1) the vertices of L are placed on a vertical line (say x=0); 2) the vertices of R are placed on another vertical line (say x=1); 3) the edges are drawing as straight-line segments. Assuming that the position of vertices of one layer is held fixed (we assume, without loss of generality, that L is the fixed layer) we consider the problem of finding the permutation of R that minimize the number of arc crossings. As shown in [5] also this problem is NP-complete. Several heuristics ([14], [2], [4], [6], [12]) have been proposed for this problem. A common characteristic of these approaches is to consider one vertex at a time, and to compute its position in the layer indipendently from the other vertices. The median heuristic [6] is the only one for which an upper bound on the error is known, i.e., the number of crossings computed is at most three times the optimum, and was shown in [4] to perform best in practice.

In this paper we present a new heuristic, called *assignment heuristic*, for minimizing arc crossings which is based on a reduction to an assignment problem [10]. This heuristic, differently from the ones mentioned above, positions simultaneously all the vertices, with the advantage of taking into account the mutual interaction of the position of all vertices in the layer. In [3] we also report on experimental results and compare the assignment heuristic with the median heuristic. For a sample of 2000 randomly generated graphs, this comparison shows that the number of crossings computed by the assignment heuristic is always less than the number produced by the other heuristics ; in particular, for density > 40%, the improvement in percentage is at least 10%.

THE ASSIGNMENT HEURISTIC

In this section we show how to reduce our problem to an assignment problem [10]. We recall that an assignment problem of size m is represented as follows:

$$Z^* = \min Z = \min \Sigma_{i=1}^{m} \Sigma_{j=1}^{m} c_{ij} x_{ij} \quad \text{with} \quad \Sigma_{i=1}^{m} x_{ij} = 1 \quad \text{and} \quad \Sigma_{j=1}^{m} x_{ij} = 1$$

where the cij's are the element of a matrix C(m×m), which fully characterizes the problem.

Let G = (L, R, E) be a 2-layered network, with $L = \{l_1,..,l_n\}$ being the fixed part. Let A(n×m) be the adjacency matrix associated with G, and B(n×m×n×m) be the matrix whose generic element B_{abcd} is 1 iff the following condition holds:

(c>a and d<b) or (c<a and d>b)

and is 0 otherwise. It is easy to see $B_{abcd} = 1$ if and only if there is a crossing between the edge (a,b), and the edge (c,d). Therefore, B represents all the possible crossings of a complete bipartite graph with $|L| = n$ and $|R| = m$.

We associate with G an assignment problem by defining the matrix C as follows:

$$c_{ij} = \Sigma_{h=1}^{n} [a_{hi} (\Sigma_{c=1}^{m} \Sigma_{d=1}^{n} B_{jhcd})]$$

Therefore, c_{ij} represents the maximum number of crossings resulting from placing a vertex i in position j. This value is computed as if the remaining part of the graph (i.e. the graph constituted by all the vertices except i) were complete.

At this point, we have reduced our problem to an assignment problem, which can be solved in polynomial time (see, for example, [10]). The solution of the assignment problem is used to obtain a permutation of the vertices of R as follows: vertex i is placed in position j if and only if $x_{ij}=1$. Notice that, since we compute c_{ij} by assuming that the remaining part of the graph is complete, the value c_{ij} represents an upper bound for the number of crossings produced by the outgoing edges from vertex i in position j. The potential error introduced by this assumption is sufficiently compensated by the idea of minimizing the function Z. In fact, the minimization of Z allows to globally consider the mutual interaction of all the vertices in the right part of G, and to partially discriminate the crossings produced by the edges of G, from those introduced by the assumption of considering a part of G complete. From this observation, it follows that our heuristic possesses a nice property: if there exists a permutation of R that produces no crossing in G, then the assignment heuristic computes this permutation.

AN OPTIMALITY CRITERION

In this section we present a method for checking whether the assignment heuristic, when applied to a graph G, determines the optimal permutation of R. First, we notice that the permutation of R generated by the solution of the assignment heuristic induces a particular permutation of the columns of the matrix A. We denote with A^* the adiacency matrix of G having correspondence to this permutation : $a_{ij}^* = \Sigma_{h=1}^{m} x_{hi} a_{hj}$

Hence, we can express Z^* as follows: $Z^* = \min Z = \Sigma_{i=1}^{m}[\,\Sigma_{h=1}^{n} a_{hi}^* (\Sigma_{c=1}^{m} \Sigma_{d=1}^{n} B_{ihcd})]$

Clearly, Z^* is not the number of crossings associated with A^*. In fact, such number, called T, is determined by the following formula: $T = {}^1/_2\, \Sigma_{i=1}^{m}[\,\Sigma_{h=1}^{n} a_{hi}^* (\Sigma_{c=1}^{m} \Sigma_{d=1}^{n} B_{ihcd}\, a_{dc}^*)]$

The most important contribution to the difference between Z^* and T is given by the terms a_{dc}^* multiplying each B_{ihcd}, whose values are either 1 or 0. Therefore, the value of Z^* can be considered as obtained by assuming the value 1 for each of such terms. This means that, while the rows of matrix B (we are referring to a 2-dimension representation of B) corresponding to edges which do not appear in G, are not considered in computing Z^*, this is not true for the columns of B. So, we can write Z^* as:

$$Z^* = 2\,\{{}^1/_2\, \Sigma_{i=1}^{m}[\,\Sigma_{h=1}^{n} a_{hi}^* (\Sigma_{c=1}^{m} \Sigma_{d=1}^{n} B_{ihcd}\, a_{dc}^*)] + {}^1/_2\, \Sigma_{i=1}^{m}[\,\Sigma_{h=1}^{n} a_{hi}^* (\Sigma_{c=1}^{m} \Sigma_{d=1}^{n} B_{ihcd}\, (1-a_{dc}^*)]\} = 2\,(T + Y)$$

where Y is:

$$Y = {}^1/_2\, \Sigma_{i=1}^{m}[\,\Sigma_{h=1}^{n} a_{hi}^* (\Sigma_{c=1}^{m} \Sigma_{d=1}^{n} B_{ihcd}\, (1-a_{dc}^*)]$$

Since Z^* is the minimum of Z, it follows that, for each permutation of the columns of $A \neq A^*$, there exists a suitable constant b such that: $T + Y \leq a+b$

where a is the number of edge crossings determined by the permutation of the columns of A.

Notice that $Z^* = MIN(Z) = MAX(N-Z)$, where N is the number of crossings of a complete graph of size (m,n). As we said before, we compute Z^* from the matrix B by considering only the minimization with respect to the rows of B, and not with respect to its columns. Therefore, if $N-Z^*$ was equal to Z^*-T, A^* would be the only permutation of the columns of A minimizing the number of crossings both with respect to the rows and to the columns of B. Indeed, this cannot happen, because the 1's of B which are eliminated in computing T, when considering the rows of B, are partially superimposed to those eliminated when considering its columns. The possible errors of the heuristic comes from such overlaps; in fact, there may exist a permutation of the columns of A, different from that of A^*, that, although corresponding to a $Z > Z^*$, produces a number of crossings smaller than T, just because it produces less overlaps. At this point, we can define a new matrix B^*, obtained from A^* and B as follows. The generic element of B^* is: $B_{ihcd}^* = a_{ih}^*\, B_{ihcd}$

Actually, B^* represents a number of crossings which is equal to Z^*. Now, let us consider the assignment obtained from A^* and TRANS(B^*), i.e. the transpose of B^*, and let T^* be its solution. Clearly, $T^* = MIN\ T = MAX(Z^* - T) = MAX\ Y = Y^*$. Since $\forall\ Z{:}\ \ Z^* = T+Y \leq Z$ and $\forall\ Y{:}\ \ Y^* \geq Y$ it follows that $\forall\ T{:}\ \ T^* \leq T$

This means that T^* is less than or equal to the minimum number of crossings, and T^* is equal to T if and only if T is the minimum number of edge crossings of G (and, therefore, A^* determines the optimum permutation of the right part of G). In the opposite case, the heuristic did not result in the optimum, and the difference between T^* and T gives an upper bound for the error.

Acknowledgements

I would like to thank Maurizio Lenzerini and Roberto Tamassia for many extremely helpful suggestions, ideas and comments. I am also grateful for the encouragement and support of Giuseppe di Battista, Carlo Batini, and Alberto Marchetti-Spaccamela.

References

[1] M.Burstein, R.Pelavin - Hierarchical Wire Routing - pp.223-234 in IEEE Trans. on CAD, October 1983.

[2] M.J.Carpano - Automatic Display of Hierarchized Graphs for Computer Aided Decision Analysis - pp. 705-715 in IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-10, 11, November 1980.

[3] T.Catarci - The Assignment Heuristic for Drawing Bipartite Graphs - Technical Report No. 10-88, Dip. di Informatica e Sistemistica, Universita' di Roma "La Sapienza", Roma, Italy, July 1988.

[4] P.Eades, D.Kelly - Heuristics for drawing 2-layered networks - pp. 89-98 in Ars Combinatoria 21-A, 1986.

[5] P.Eades, B.McKay, N.Wormald - An NP-complete crossing number problem for bipartite graphs - Technical Report No. 60, Dept. of Computer Science, University of Queensland, St.Lucia, April 1985.

[6] P.Eades, N.Wormald - The median heuristic for drawing 2-layered networks - Technical Report No. 69, Dept. of Computer Science, University of Queensland, St.Lucia, May 1986.

[7] M.R.Garey, D.S.Johnson - Crossing number is NP-complete - pp. 312-316 in SIAM J. Algebraic Discrete Methods 4, 3, 1983.

[8] T.C.Hu, E.Kuh (eds.) - VLSI Circuit Layout - IEEE Press, 1985.

[9] C.P.Hsu - Minimum-Via Topological Routing - pp.235-246 in IEEE Trans. on CAD, October 1983.

[10] H. W. Kuhn - The Hungarian Method for the Assignment Problem - pp. 83 - 97 in Novae Research Logistics Quarterly, 2, 1955.

[11] F.T.Leighton - New Lower Bound Techniques for VLSI - pp.47-70 in Math. Systems Theory , 17, 1984.

[12] L.A.Rowe, M.Davis, E.Messinger, C.Meyer, C.Spirakis, A.Tuan - A Browser for Directed Graphs - pp.61-67 in Practice and Experience Vol. 17, 1, 1987.

[13] C.D.Thompson - Area-time complexity for VLSI - pp.81-88 in Proc.of the 11th Annual Symposium on Theory of Computing, May 1979.

[14] J.N. Warfield - Crossing theory and hierarchy mapping - pp. 505-523 in IEEE Trans. Syst. Man Cybern. SMC-7, 7, 1977.

A STATE-VARIABLE APPROACH TO THE MODELING AND CONTROL OF DISCRETE-EVENT SYSTEMS

Y. Li and W.M. Wonham
Systems Control Group
Dept. of Electrical Engineering
University of Toronto
Toronto, Canada M5S 1A4

ABSTRACT

A compact model for discrete-event systems (DES) is proposed. The state space of a DES consists of "vectors" generated by a finite number of state variables ranging over the integers. Concurrency is incorporated in the model. The state-feedback control of such DES is studied.

1. INTRODUCTION

Discrete-event systems (DES) can be described as those which are discrete (in time and state space), asynchronous and typically nondeterministic. Examples include flexible manufacturing systems, traffic control systems and many computer and communication systems. A control theory of discrete-event systems was initiated by Ramadge and Wonham [9], in which a DES is modeled by a controlled state-transition system. The modular feedback logic for DES was studied in [8]. A further extension of [8] can be found in [6], where controllability and observability were introduced into the state-feedback control of DES. In this paper we work in the same basic framework, but exploit the fact that in many applications the DES state description will admit arithmetic or algebraic structure. This will be true, for instance, when the overall system includes buffers, counters, or groups of similar entities (e.g. machines) where in each group only the number of entities in a given state is relevant to the problem description.

While a long-standing technique for the description of DES with arithmetic structure is provided by Petri nets [7], only recently have control structure and control synthesis problems of RW type been explicitly introduced into the Petri net framework[3]. By contrast, in the spirit of standard control theory, we prefer to set up a natural state space, in this case a space of "vectors" over the integers, and develop a specialized version of RW control theory from scratch. Our use of Petri nets will be confined to the graphical description of examples. The model proposed in this paper is similar to the vector addition system of Karp and Miller [1] and the vector replacement system of Keller [2]. However, in our model we distinguish between controllable and uncontrollable events; furthermore, we focus on quantitative features of various control-theoretic problems.

Concurrency is an issue arising frequently in the design of computer operating systems, data base management systems and the verification of parallel programs [1,2]; it is a phenomenon in a system associated with the independent actions of its subsystems. When it comes to modeling concurrent systems, there exist two philosophies in the computer community. One holds that the behavior of a concurrrent system is just the interleaving of the events of its subsystems, while the other maintains that the behavior of a concurrent system consists of not only the interleaving but also the simultaneous events of its subsystems. In what follows, the concurrency without simultaneity will be called *weak concurrency* while that with simultaneity *strict concurrency*. From a control-theoretic point of view, the inclusion of simultaneity in the model is necessary. Because of the importance of simultaneity, entire Section 5 will be devoted to its treatment. Until that point, strict simultaneity will be ruled out. Proofs omitted in this conference paper will be documented in [4].

2. MODELING OF DISCRETE-EVENT SYSTEMS

Suppose that the state of a DES **G** is described by *state variables* $\mathbf{x}_1, \mathbf{x}_2, \ldots$ and $\mathbf{x}_n$, ranging over the integers **Z**. The state space of **G** is then

$$\mathbf{S} := \{ X = [x_1, x_2, \ldots, x_n] \mid x_i \in \mathbf{Z} \}$$

Let the *event set* of **G** be Σ. For each $\alpha \in \Sigma$ the *occurrence condition* C_α is a predicate on **S** specified by

$$X \geq J_\alpha$$

where $J_\alpha \in \mathbf{S}$ with nonnegative entries. The state transition function of **G** associated with $\alpha \in \Sigma$ is $\delta_\alpha : \mathbf{S} \to \mathbf{S}$ specified by

$$\delta_\alpha(X) = X + I_\alpha$$

We write "$\delta_\alpha(X)!$" to mean "α can occur at X", i.e. $X \in C_\alpha$. Only a mild condition on the transition function is imposed in our model:

$$(\forall X)\, X \geq J_\alpha \Rightarrow X + I_\alpha \geq 0$$

which guarantees that the system has nonnegative state components when starting with nonnegative initial state components. If we denote the set of all state transitions by δ_Σ and the set of all occurrence conditions by C_Σ, then **G** is described by a 5-tuple: $\mathbf{G} := (\Sigma, \mathbf{S}, \delta_\Sigma, C_\Sigma, X_0)$, where X_0 is the initial state.

Let δ: $\Sigma^* \times \mathbf{S} \to \mathbf{S}$ be the extension of δ: $\Sigma \times \mathbf{S} \to \mathbf{S}$. Define the *reachable state set* $\mathbf{R}_X$ of a state $X \in \mathbf{S}$ as

$$\mathbf{R}_X := \{ \delta_w(X) \mid w \in \Sigma^*, \delta_w(X)! \}$$

and define the *feasible event-string set* $\mathbf{L}_X$ of X as

$$\mathbf{L}_X := \{ w \in \Sigma^* \mid \delta_w(X)! \}$$

If we identify places and transitions in Petri nets with state variables and events respectively, then the above model is similar to a Petri net but with less restrictive occurrence conditions for events. Moreover, we shall take an algebraic rather than net-theoretic approach to the analysis of the model dynamics, and make no appeal to net-theoretic methods. In the following, we assume that each entry of J_α is either 0 or 1, and that each entry of I_α is either 0, 1, or -1. The results can easily be extended to the general case. Under this assumption we introduce the following definitions.

Definition 2.1: For an event $\alpha \in \Sigma$, define relation "$\to_\alpha$" on the set of state variables by:

$$\mathbf{x}_i \to_\alpha \mathbf{x}_j \quad \text{if } J_\alpha(i) = 1 \text{ and } I_\alpha(j) = 1.$$

In words, "$\mathbf{x}_i \to_\alpha \mathbf{x}_j$" means that $\mathbf{x}_i$ and $\mathbf{x}_j$ are related via α in the following way: the value of $\mathbf{x}_i$ must be greater than or equal to 1 for α to occur, and the occurrence of α will increase the value of $\mathbf{x}_j$ by 1.

Definition 2.2:

(1) For an event $\alpha \in \Sigma$, define the following sets:

$$\alpha^+ := \{ \mathbf{x}_i \mid I_\alpha(i) = 1 \},\ \alpha^- := \{ \mathbf{x}_i \mid I_\alpha(i) = -1 \},\ \alpha^\# := \{ \mathbf{x}_i \mid J_\alpha(i) = 1 \}.$$

(2) For a state variable $\mathbf{x}_i$, define the following sets:

$$\mathbf{x}_i^+ := \{ \alpha \mid I_\alpha(i) = -1 \},\ \mathbf{x}_i^- := \{ \alpha \mid I_\alpha(i) = 1 \}.$$

By this definition, α is permitted to occur only if the values of state variables in $\alpha^\#$ are all greater than or equal to 1, and after α occurs the values of the state variables in α^+ (α^-) will be increased (decreased) by 1; the occurrence of an event in $\mathbf{x}_i^+$ ($\mathbf{x}_i^-$) will decrease (increase) the value of $\mathbf{x}_i$ by 1. The two concepts introduced below play an important role in our later discussion.

Definition 2.3: For a state variable $\mathbf{x}$, define two sets $\Omega_{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ inductively as follows:

(1) $\mathbf{x} \in \Omega_{\mathbf{x}}$;

(2) If $\mathbf{x}_i \rightarrow_\alpha \mathbf{x}_j$ for some $\mathbf{x}_j \in \Omega_{\mathbf{x}}$ then $\mathbf{x}_i \in \Omega_{\mathbf{x}}$ and $\alpha \in \Sigma_{\mathbf{x}}$;

(3) Every element of $\Omega_{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ is obtained as in (1) or (2).

Definition 2.4: For an event α, define two sets Ω_α and Σ_α inductively as follows:

(1) $\alpha \in \Sigma_\alpha$;

(2) If $\mathbf{x}_i \rightarrow_\beta \mathbf{x}_j$ for some $\beta \in \Sigma_\alpha$ and some $\mathbf{x}_j$ then $\mathbf{x}_i \in \Omega_\alpha$

If $\mathbf{x}_i \rightarrow_\beta \mathbf{x}_j$ for some $\mathbf{x}_j \in \Omega_\alpha$ and some $\mathbf{x}_i$ then $\beta \in \Sigma_\alpha$;

(3) Every element of Ω_α and Σ_α is obtained as in (1) or (2).

Briefly speaking, $\Omega_{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ (or, Ω_α and Σ_α) include all state variables and events which can "influence" $\mathbf{x}$ (or, α) respectively. Later on, we shall encounter the algebraic operators defined below.

Definition 2.5:

(1) Let P be a predicate on S. Define the *natural projection* of P, $T_{\mathbf{x}_i}(P)$, by

$$T_{\mathbf{x}_i}(P) := \{ X(i) \mid X \in P \}$$

For a state variable set $s = \{ \mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_j} \}$ with $i_1 < i_2 < \ldots < i_j$ define the *natural projection* of P, $T_s(P)$, by

$$T_s(P) := \{ [X(i_1), X(i_2), \ldots, X(i_j)] \mid X \in P \};$$

(2) Let $L \subseteq \Sigma^*$. For an event subset $\Sigma_0 \subseteq \Sigma$, define the *natural projection* of L, $Y_{\Sigma_0}(L)$, to be the set of all strings in Σ_0^* obtained by erasing events in $\Sigma - \Sigma_0$ from strings of L;

(3) Let $L_1, L_2 \subseteq \Sigma^*$. The *shuffle* $L_1 \| L_2$ of L_1 and L_2 is defined as

$$L_1 \| L_2 := \{ w \in \Sigma^* \mid w \text{ is an interleaving of a string } w_1 \in L_1 \text{ and a string } w_2 \in L_2 \}.$$

The natural projection above allows us to focus on "local" dynamic behavior of the model; the shuffle operator captures a notion of decoupledness at the language level. In the next section we discuss state-feedback control with the model just proposed. To facilitate presentation we shall write "$X \in P$" to denote "$P(X) = 1$" for a predicate P.

3. STATE-FEEDBACK CONTROL OF DISCRETE-EVENT SYSTEMS

In a DES $\mathbf{G} = (\Sigma, \delta_\Sigma, C_\Sigma, X_0)$ as described in the previous section, a control mechanism is adjoined in the following way. We first designate a subset $\Sigma_c \subseteq \Sigma$ of *controllable events*, and call $\Sigma_u = \Sigma - \Sigma_c$ the *uncontrollable event* subset. Then the set of *control patterns* for G is defined to be

$$\Gamma := \{ \gamma \mid \gamma : \Sigma \rightarrow \{0,1\} \;\&\; \gamma(\alpha) = 1 \text{ for each } \alpha \in \Sigma_u \}$$

We say that $\alpha \in \Sigma$ is *enabled* by γ if $\gamma(\alpha) = 1$, and *disabled* by γ if $\gamma(\alpha) = 0$. A *(static) state-feedback controller* for G is a function f: S$\rightarrow\Gamma$. Denote by S(f/G) the set of all states reachable from the initial state X_0 in the closed-loop system f/G. Then it was shown in [6] that a predicate P on S can be synthesized by a controller f (i.e. $P = \mathrm{S}(f/\mathrm{G})$) if and only if P is *controllable*, or formally if and only if

$$P \le \mathrm{Re}(\mathbf{G}, P) \wedge wlp_{\Sigma_u}(P)$$

where Re(G,P) is the set of all states reachable from X_0 via states X satisfying P (i.e. $X \in P$), and $wlp_{\Sigma_u}(P)$ is defined to be the conjunction of all $wlp_\alpha(P)$, $\alpha \in \Sigma_u$ with

$$wlp_\alpha(P)(X) := \begin{cases} 1 & \text{if } X \in C_\alpha \text{ and } \delta_\alpha(X) \in P, \textit{ or } \text{if } X \notin C_\alpha \\ 0 & \textit{otherwise} \end{cases}$$

Controllability is a strong version of *control-invariance* in [8]. A predicate P is control-invariant if $P \leq wlp_{\Sigma_u}(P)$. A given predicate P that specifies legal states is usually not controllable. This leads us to approximate P with a weakest controllable subpredicate. It can be shown that the set of all controllable subpredicates is a semilattice with respect to set inclusion, and a supremal element in the set exists [6]. Denote this element by $supC(P)$. Define a controller f^* by

$$f_\alpha^* := wlp_\alpha(supC(P)) \quad (rel\ P)$$

Then it can be proved that $\mathrm{S}(f^*/\mathbf{G}) = supC(P)$. This controller is optimal in the sense that for any controller f with $\mathrm{S}(f/\mathbf{G}) \leq P$ we have $\mathrm{S}(f/\mathbf{G}) \leq \mathrm{S}(f^*/\mathbf{G})$. The action of this optimal controller is interpreted as follows: Enable a controllable event α at a state X if and only if $\delta_\alpha(X) \in supC(P)$. Therefore, the construction of the optimal state-feedback controller amounts to the computation of the supremal controllable subpredicate of P.

Now define $[P]$ to be the set of all states $X \in \mathrm{S}$ satisfying

$$(\forall w \in \Sigma_u^*) \quad \delta_w(X) \in P$$

Theorem 3.1: Assume that $X_0 \in P$. If $X_0 \notin [P]$ then $supC(P) = \varnothing$; otherwise, $supC(P) = \mathrm{Re}(\mathbf{G}, [P]) \wedge [P]$.

By virtue of Theorem 3.1, the action of the optimal controller f^* based on $supC(P)$ can be further interpreted as: Starting at the initial state, enable a controllable event α at X if and only if $\delta_\alpha(X) \in [P]$. So, from the viewpoint of controller synthesis, the computation of $supC(P)$ can be reduced to that of $[P]$, which depends only on the dynamics of the uncontrollable part of $\mathbf{G}$, denoted by $\mathbf{G}_u$. Formally, $\mathbf{G}_u := (\Sigma, \mathrm{S}, \delta_{\Sigma_u}, C_{\Sigma_u}, X_0)$, where δ_{Σ_u} and C_{Σ_u} are restrictions of δ_Σ and C_Σ to Σ_u respectively.

4. COMPUTATION OF SupC(P)

As mentioned in the last section, the supremal controllable subpredicate $supC(P)$ equals $Re(\mathbf{G}, [P]) \wedge [P]$ (if $X_0 \in [P]$) and for the purpose of the optimal controller synthesis, the computation of $supC(P)$ is reduced to that of $[P]$. In this section we focus on the computation of $[P]$ for a class of "linear" legal-state specifications P given by: $AX \leq b$, where $A = [a_1, a_2, \ldots, a_n]$, $a_i \in \mathbf{Z}$ and $b \in \mathbf{Z}$. These "linear" predicates alone are sufficient to express specifications such as "safety" and "fairness", etc. (cf. Section 6), and can be combined by logical connectives to obtain more complex predicates whose supremal controllable subpredicates can be computed modularly; for instance, the following is true:

$$supC(P_1 \wedge P_2) = supC(P_1) \wedge supC(P_2)$$

Since the computation of $[P]$ depends only on the dynamics of the uncontrollable part of $\mathbf{G}$, we assume in this section that the notations α^+, α^-, $\alpha^\#$, $\mathbf{x}^+$, $\mathbf{x}^-$ and $\mathbf{L}_X$, $\mathbf{R}_X$, $\Omega_\mathbf{x}$, Σ_α etc. are all defined for $\mathbf{G}_u$.

For the given A and b, a state $\hat{X} \in [P]$ if and only if

$$(\forall X \in \mathbf{R}_{\hat{X}}) \quad AX \leq b$$

or equivalently,

$$\max\{a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \mid X = [x_1, x_1, \ldots, x_n] \in \mathbf{R}_{\hat{X}}\} \leq b$$

We now discuss how to compute this maximum. To simplify notation we write $\mathbf{R}$, $\mathbf{L}$ in place of $\mathbf{R}_{\hat{X}}$ and $\mathbf{L}_{\hat{X}}$. For the purpose of illustration, we shall focus on the case when $A = [a_1, a_2, a_3, 0, \ldots, 0]$ with a_1, $a_2 > 0$, $a_3 < 0$; the general case can be treated similarly. For an arbitrary $\mathbf{G}_u$, the computation of $\max\{a_1 x_1 + a_2 x_2 + a_3 x_3 \mid X \in \mathbf{R}\}$ is usually a nonlinear programming problem with an unstructured feasible solution set $\mathbf{R}$, and thus not subject to attack by standard optimization techniques. However, as will be shown in this section, the problem can be solved easily by exploiting special structures of $\mathbf{G}_u$. We consider one such special structure in this section. Basically, it will be shown that when $\mathbf{G}_u$ has some appropriately decoupled structure, the computation of the above maximum can be carried through recursively, and each recursion is arithmeticaly simple. Results of this section are given in a series of claims. Those results per se are not surprising, but their formal proofs turn out to be involved. The difficult problem encountered is how to analyse the (weak) concurrency in the model. We start with a definition of decoupledness:

Definition 4.1: Two event subsets Σ_1, $\Sigma_2 \subseteq \Sigma$ are *decoupled* with respect to $\mathbf{L} \subseteq \Sigma^*$, denoted by $\Sigma_1 \mid \Sigma_2$, if

$$(\mathbf{L} \cap \Sigma_1^*) \,\|\, (\mathbf{L} \cap \Sigma_2^*) = \mathbf{L} \cap (\Sigma_1 \cup \Sigma_2)^*.$$

Thus, $\Sigma_1 \mid \Sigma_2$ implies that the system dynamics associated with Σ_1 and Σ_2 are "independent" of each other. This definition can be extended to the case of more than two event subsets.

We use the following two assumptions to characterize two intuitively-correct properties of our model dynamics, on which our later conclusions will be based. The first assumption states that for any two state variables $\mathbf{x}_i$ and $\mathbf{x}_j$ (or, two events α_1 and α_2), if $\Sigma_{\mathbf{x}_i}$ and $\Sigma_{\mathbf{x}_j}$ (or, Σ_{α_1} and Σ_{α_2}) are disjoint then the system dynamics associated with these two subsets of events are "independent". The second assumption states that if w is a feasible event string then the projection of w on the event subset $\Sigma_{\mathbf{x}}$ (or, Σ_α) is again a feasible event string; that is to say, the system dynamics associated with $\Sigma_{\mathbf{x}}$ (or, Σ_α) cannot be influenced by events other than those in $\Sigma_{\mathbf{x}}$ (or, Σ_α).

Assumption 4.1: For two state variables $\mathbf{x}_i$ and $\mathbf{x}_j$ (or, two events α_1 and α_2), if $\Sigma_{\mathbf{x}_i} \cap \Sigma_{\mathbf{x}_j} = \varnothing$ (or, $\Sigma_{\alpha_1} \cap \Sigma_{\alpha_2} = \varnothing$) then $\Sigma_{\mathbf{x}_i} \mid \Sigma_{\mathbf{x}_j}$ (or, $\Sigma_{\alpha_1} \mid \Sigma_{\alpha_2}$).

Assumption 4.2: For any state variable $\mathbf{x}$ and any event α we have

$$(\forall w \in \mathbf{L})\ Y_{\Sigma_{\mathbf{x}}}(w) \in \mathbf{L} \quad \text{and} \quad (\forall w \in \mathbf{L})\ Y_{\Sigma_\alpha}(w) \in \mathbf{L}.$$

Claim 4.1: For a state variable $\mathbf{x}$, if $\mathbf{x}^+ \cap \Sigma_{\mathbf{x}} = \varnothing$ then $Y_{\mathbf{x}^-}(\mathbf{L} \cap (\Sigma - \mathbf{x}^+)^*) = Y_{\mathbf{x}^-}(\mathbf{L})$.

This claim asserts that if no event in $\mathbf{x}^+$ appears in $\Sigma_{\mathbf{x}}$, then the occurrences of events in $\mathbf{x}^-$ will not "depend" on those of events in $\mathbf{x}^+$. The following claim asserts that if no event in $\mathbf{x}^+$ appears in $\Sigma_{\mathbf{x}}$ then the maximal $\mathbf{x}$ is achieved after the occurrence of an event string which has a maximal number of occurrences of events in $\mathbf{x}^-$ among all feasible event strings.

Claim 4.2: For a state variable $\mathbf{x}$, if $\mathbf{x}^+ \cap \Sigma_{\mathbf{x}} = \varnothing$ then $\max T_{\mathbf{x}}(\mathbf{R}) = \hat{x} + \max\{\, |w|_{\mathbf{x}^-} \mid w \in \mathbf{L} \}$, where $|w|_{\mathbf{x}^-}$ denotes the number of appearances in w of events in $\mathbf{x}^-$.

The following claim reduces the computation of $\max\{\, |w|_{\mathbf{x}^-} \mid w \in \mathbf{L} \}$ to that of $\max\{\, |w|_{\alpha_i} \mid w \in \mathbf{L} \}$ for $\alpha_i \in \mathbf{x}^-$.

Claim 4.3: Let $\mathbf{x}^- = \{ \alpha_1, \alpha_2, ..., \alpha_p \}$ for some state variable $\mathbf{x}$. If Σ_{α_1}, Σ_{α_2}, ..., and Σ_{α_p} are pairwise disjoint, then

$$\begin{aligned} &\max\{\, |w|_{\mathbf{x}^-} \mid w \in \mathbf{L} \} \\ &\quad = \max\{\, |w|_{\alpha_1} \mid w \in \mathbf{L} \} + \max\{\, |w|_{\alpha_2} \mid w \in \mathbf{L} \} + ... + \max\{\, |w|_{\alpha_p} \mid w \in \mathbf{L} \}. \end{aligned}$$

Claim 4.4: Let $\alpha^- = \{ \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p \}$, $\alpha^\# = \{ \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k \}$. Suppose that $\Sigma_{\mathbf{z}_1}, \Sigma_{\mathbf{z}_2}, ...,$ and $\Sigma_{\mathbf{z}_k}$ are pairwise disjoint. If $\min\{ \max T_{\mathbf{z}_i}(\mathbf{R}) \mid i = 1, 2, ..., k \} = 0$ then $\max\{\, |w|_\alpha \mid w \in \mathbf{L} \} = 0$; otherwise,

$$\max\{\, |w|_\alpha \mid w \in \mathbf{L} \} = \min\{ \max T_{\mathbf{v}_i}(\mathbf{R}) \mid i = 1, 2, ..., p \}.$$

For the purpose of illustration, we assume that $\mathbf{x}_3^+ = \{ \alpha \}$ and $\alpha^- = \alpha^\# = \{ \mathbf{x}_3, \mathbf{x} \}$; the treatment of the general case follows easily.

Claim 4.5: If the following two conditions hold:

(1) $\mathbf{x}^+ \cap \Sigma_{\mathbf{x}} = \varnothing$, $\mathbf{x}_3^+ \cap \Sigma_{\mathbf{x}_3} = \varnothing$

(2) $\Sigma_{\mathbf{x}} \cap \Sigma_{\mathbf{x}_3} = \varnothing$

then

$$\min T_{\mathbf{x}_3}(\mathbf{R}) = \begin{cases} \hat{x}_3 - \max T_{\mathbf{x}}(\mathbf{R}) & \text{if } \max T_{\mathbf{x}}(\mathbf{R}) < \hat{x}_3 \\ 0 & \textit{otherwise} \end{cases}$$

Finally, we have the following claim which gives conditions under which $\max\{ a_1x_1+a_2x_2-|a_3|x_3 \mid X\in \mathbf{R} \}$, thus $[P]$, can be computed using the previous claims.

Claim 4.6: If the following two conditions hold:

(1) $\mathbf{x}_1^{\dagger}\cap\Sigma_{\mathbf{x}_1} = \varnothing, \mathbf{x}_2^{\dagger}\cap\Sigma_{\mathbf{x}_2} = \varnothing, \mathbf{x}_3^{\dagger}\cap\Sigma_{\mathbf{x}_3} = \varnothing$

(2) $\Sigma_{\mathbf{x}_1}$, $\Sigma_{\mathbf{x}_2}$, and Σ_{α} are pairwise disjoint

then

$$\max\{ a_1x_1+a_2x_2-|a_3|x_3 \mid X\in \mathbf{R} \} = a_1\max T_{\mathbf{x}_1}(\mathbf{R})+a_2\max T_{\mathbf{x}_2}(\mathbf{R})-|a_3|\min T_{\mathbf{x}_3}(\mathbf{R}).$$

The previous claims are combined to compute $\max\{ a_1x_1+a_2x_2+a_3x_3 \mid X\in \mathbf{R} \}$ when the model has structure as specified in them: Claim 4.6 reduces the computation of this maximum to $\max T_{\mathbf{x}_1}(\mathbf{R})$, $\max T_{\mathbf{x}_2}(\mathbf{R})$, and $\min T_{\mathbf{x}_3}(\mathbf{R})$; $\max T_{\mathbf{x}_1}(\mathbf{R})$ and $\max T_{\mathbf{x}_2}(\mathbf{R})$ are computed by applying Claim 4.2, 4.3, 4.4; $\min T_{\mathbf{x}_3}(\mathbf{R})$ is computed using Claim 4.5. For instance, in the case of Petri nets, when the subnets rooted at x_1, x_2 and x_3 are decoupled and have a "tree-like" structure we apply the above claims to recursively compute this maximum -- starting at leaves and going up to the roots. The computation needed is simple to perform (cf. Section 6). The "tree-like" structure requirement will be satisfied by many DESs which have a relatively small number of uncontrollable events or have losely "coupled" uncontrollable events. Also, some more complicated structures can be based on the "tree-like" structure and the results obtained in this section can be used as a basis for the analysis of these new structures.

5. STRICT CONCURRENCY IN THE MODEL

Up to now we have been assuming that events in the model occur in an interleaving fashion and simultaneity of events is ruled out, i.e. we only allowed weak concurrency; this simplifies our discussion. Even though the existence of "exactly" simultaneous events is debatable philosophically, the inclusion of simultaneity in our model does have practical importance: When events occur "almost" at the same time they can be considered as simultaneous events from a controller's point of view, because any physical sensor connected to a controller has limited temporal resolution. In this section we extend our previous model to include simultaneity, i.e., to allow strict concurrency.

Define the event set of the extended model by $\Sigma_{con} := 2^{\Sigma}$. An event $e=\{\alpha_1,\alpha_2,\ldots,\alpha_m\}\in\Sigma_{con}$ represents the simultaneous occurrence of α_1, α_2, ..., and α_m. By identifying event $\{\alpha\}$ in Σ_{con} with α, we have $\Sigma\subseteq\Sigma_{con}$ with Σ_{con} being usually much larger than Σ. An event $e\in\Sigma_{con}$ is allowed to occur at a state X if every $\alpha\in e$ can occur at X and the simultaneous occurrence of all α in e will not result in a negative state component. Formally, we define the occurrence condition C_e of e by: $X\in C_e$ if

(1) $(\forall\alpha\in e)\ X\in C_{\alpha}$, and

(2) $(\forall\alpha\in e)(\forall \mathbf{x}_i\in\alpha^{-})\ x_i \geq |\mathbf{x}_i^{\dagger}\cap e|$

where $|.|$ denotes set cardinality. Let $con(X) := \{ e \mid X\in C_e \}$, and let $v(e)$ be the set of concatenations of all events in e. Define the transition function $\delta_e(X)$ for events $e\in con(X)$ by: $\delta_e(X) := \delta_w(X)$, where $w\in v(e)$. It can be verified that this function is well-defined for our model; that is,

$$(\forall w_1, w_2\in v(e))\ \delta_{w_1} = \delta_{w_2}$$

Denote by $\delta_{\Sigma_{con}}$ the set of all transition functions δ_e and by $C_{\Sigma_{con}}$ the set of all occurrence conditions. Our extended model is then described by a 5-tuple: $\mathbf{G}_{con} := \{ \Sigma_{con}, \mathrm{S}, \delta_{\Sigma_{con}}, C_{\Sigma_{con}}, X_0 \}$. This model has richer dynamics than $\mathbf{G}$ owing to the presence of simultaneity, and it poses additional problems for its control. The set of events which can be directly controlled by a controller is Σ_c, a subset of Σ_{con}; the remaining events in Σ_{con} are either uncontrollable or can only be controlled indirectly by disabling at least one of the controllable elements in them. This dependence of disablement makes it impossible to directly solve the controller synthesis problem for $\mathbf{G}_{con}$ (i.e., for a predicate P, find a controller f such that $S(f/\mathbf{G}_{con})\leq P$) by standard methods. In the following, however, we shall propose a stronger controller synthesis problem for

G and show that a solution to this problem is also a solution to the controller synthesis problem for $\mathbf{G}_{con}$. To this end, the following definition is given:

Definition 5.1: A predicate P is called *concurrently well-posed* (CWP) if the following is true:

$$(\forall X \in P)\,(\forall \{\alpha,\beta\} \in con\,(X)) \quad \delta_\alpha(X) \in P \,\wedge\, \delta_\beta(X) \in P \,\Rightarrow\, \delta_{\alpha\beta}(X) \in P.$$

Thus, at any state X satisfying a CWP predicate P, if $\{\alpha,\beta\}$ can occur in $\mathbf{G}_{con}$ and both α and β bring $\mathbf{G}_{con}$ from X to states satisfying P then, after the occurrence of event string $\alpha\beta$ at X, $\mathbf{G}_{con}$ also transits to a state satisfying P. The following theorem asserts that CWP together with controllability is a sufficient condition to guarantee that there exists a controller which synthesizes P as the closed-loop predicate in both G and $\mathbf{G}_{con}$ (i.e. concurrency has no effect on control).

Theorem 5.1: If P is controllable and CWP then for the controller f^* specified by

$$f^*_\alpha := wlp_\alpha(P) \;\; (rel\ P)$$

the following holds:

$$S\,(f^*/\mathbf{G}) = S(f^*/\mathbf{G}_{con}) = P.$$

Therefore, when P is both controllable and CWP the construction of a controller synthesizing P in $\mathbf{G}_{con}$ is reduced to that of a controller synthesizing P in G. If there exists a supremal element in the set of all controllable and CWP subpredicates of a given predicate specifying legal states, an ''optimal'' solution can be found to the controller synthesis problem for $\mathbf{G}_{con}$. However, such a supremal element may not exist [4]. Nevertheless, we can still find an algorithm to compute a controllable and CWP subpredicate (not necessarily optimal) as we did for the controllable and well-posed sublanguage in [5]. Of course, if the supremal controllable subpredicate as computed in the previous section turns out to be CWP, then the optimal controller constructed for G is also a solution to the controller synthesis problem in $\mathbf{G}_{con}$ (cf. Section 6).

6. EXAMPLE

We consider a flexible manufacturing system consisting of two robot-groups and a conveyor (Fig. 6.1). Both groups of robots consume the same type of components stored in their respective buffers. The components in Buffer 2 are transported from the warehouse by an automatic carrier, and those in Buffer 1 are transported from Buffer 2 via a conveyor. The Petri-net-like model of this system is shown in Fig. 6.2. From this model we can write down J_{α_i} and I_{α_i} directly according to the interpretation of Petri nets and the following convention: When a state variable $\mathbf{x}$ is connected to an event α by dotted line, we mean that $\mathbf{x} \in \alpha^\# - \alpha^-$.

There are two objectives for the control of this system. First, we require that the total number of components on the workfloor does not exceed a positive integer D; this is a safety requirement. Secondly, we require that the number of parts produced by Group 2 does not exceed that of Group 1 by a positive integer B; this is a fairness requirement: Group 2 has the first opportunity in taking components transported from the warehouse, and we want to ensure that Group 1 and Group 2 have a fair share of those components. Formally, our specification is: $P = P_1 \wedge P_2$, where $P_1 := (\mathbf{x}_{21} + \mathbf{x}_{22} \le D)$, $P_2 := (\mathbf{x}_{72} - \mathbf{x}_{71} \le B)$. By modularity, $supC\,(P) = supC\,(P_1) \wedge supC\,(P_2)$. So we can compute $supC\,(P_1)$ and $supC\,(P_2)$ separately to get $supC\,(P)$ and, ultimately, an optimal controller for the model G. To this end, only $[P_1]$ and $[P_2]$ need to be computed (Theorem 3.1).

(1) Computation of $[P_1](\hat{X})$: From the uncontrollable part of G shown in Fig. 6.3 we can easily obtain $\Omega_{\mathbf{x}_{2i}}$, $\Sigma_{\mathbf{x}_{2i}}$, as well as $\mathbf{x}^+_{2i}$ $(i=1,2)$ (Fig. 6.4). Since $\mathbf{x}^+_{2i} \cap \Sigma_{\mathbf{x}_{2i}} = \emptyset$ $(i=1,2)$ and $\Sigma_{\mathbf{x}_{21}} \cap \Sigma_{\mathbf{x}_{22}} = \emptyset$, we have

$$\max\{\mathbf{x}_{22} + \mathbf{x}_{21} \mid X \in \mathbf{R}\} = \max T_{\mathbf{x}_{22}}(\mathbf{R}) + \max T_{\mathbf{x}_{21}}(\mathbf{R})$$

from Claim 4.6. By Claim 4.2, 4.3, 4.4,

$$\max T_{\mathbf{x}_{22}}(\mathbf{R}) = \hat{x}_{22} + \max T_{\mathbf{x}_{52}}(\mathbf{R}) = \hat{x}_{22} + \hat{x}_{52} \quad \text{and} \quad \max T_{\mathbf{x}_{21}}(\mathbf{R}) = \hat{x}_{21} + \max T_{\mathbf{x}_{51}}(\mathbf{R}) = \hat{x}_{21} + \hat{x}_{51}$$

In summary, we have

$$\max\{\mathbf{x}_{22} + \mathbf{x}_{21} \mid X \in \mathbf{R}\} = \hat{x}_{22} + \hat{x}_{21} + \hat{x}_{52} + \hat{x}_{51}; \quad \text{and} \quad [P_1](\hat{X}) = (\hat{x}_{22} + \hat{x}_{21} + \hat{x}_{52} + \hat{x}_{51} \le D)$$

(2) Computation of $[P_2](\hat{X})$: From $\mathbf{G}_u$ it can be easily seen that $\mathbf{x}_{71}^{+} = \varnothing$, $\mathbf{x}_{72}^{+} \cap \Sigma_{\mathbf{x}_{72}} = \varnothing$, and $\Sigma_{\mathbf{x}_{72}} \cap \Sigma_{\mathbf{x}_{71}} = \varnothing$. So by Claim 4.6, $\max\{\mathbf{x}_{72} - \mathbf{x}_{71} \mid X \in \mathbf{R}\} = \max T_{\mathbf{x}_{72}}(\mathbf{R}) - \min T_{\mathbf{x}_{71}}(\mathbf{R})$. But

$$\max T_{\mathbf{x}_{72}}(\mathbf{R}) = \hat{x}_{72} + \max T_{\mathbf{x}_{52}}(\mathbf{R}) = \hat{x}_{72} + \hat{x}_{52}$$

from Claim 4.2, 4.3, 4.4. Also, $\min T_{\mathbf{x}_{71}}(\mathbf{R}) = \hat{x}_{71}$. So, $\max\{x_{72} - x_{71} \mid X \in \mathbf{R}\} = \hat{x}_{72} + \hat{x}_{52} - \hat{x}_{71}$ and

$$[P_2](\hat{X}) = (\hat{x}_{72} + \hat{x}_{52} - \hat{x}_{71} \le B)$$

It is not hard to check that $supC(P)$ computed as above is CWP and for the optimal controller f constructed from $supC(P)$ we have $S(f/\mathbf{G}) = S(f/\mathbf{G}_{con}) = supC(P)$, i.e., the concurrency in the system has no effect on the control. This controller's function can be paraphrased as: Disable a controllable event at a state X if and only if the event will cause the system to transit to a state $\hat{X}$ such that $\hat{x}_{22} + \hat{x}_{21} + \hat{x}_{52} + \hat{x}_{51} > D$ or $\hat{x}_{72} + \hat{x}_{52} - \hat{x}_{71} > B$.

7. CONCLUSION

A compact model for DES is studied wherein state variables range over integers; the state transition function is a function of these state variables. Our treatment is essentially algebraic, even though we do use net notation to facilitate presentation of examples. Using the results obtained, we can show that when the uncontrollable part of model has a "tree-like" structure, the computation of supremal subpredicate is arithmetically simple to perform. This structure can be used as a basis to construct further complex structures. Strict concurrency is supported; the concept of concurrent well-posedness is introduced to study its effect on control.

8. REFERENCES

1. R. Karp and R. Miller, " Parallel program schemata, " *Journal of Computer and System Science*, 3(4), 1969, pp. 167-195.

2. R. Keller, " Formal verification of parallel programs ," *Communication of the ACM*, 19(7), 1976, pp. 371-384.

3. B.H. Krogh, " Controlled Petri nets and maximally permissive feedback logic, " *Proc. 25th Annual Allerton Conference*, September, 1987.

4. Y. Li and W.M. Wonham, " A state-variable approach to the modeling and control of discrete-event systems, " Systems Control Group Report #8807, Dept. of Electrical Eng., Univ. of Toronto, September, 1988.

5. Y. Li , " On supervisory control of real-time discrete-event systems, " M.A.Sc. Thesis, Dept. of Electrical Eng., Univ. of Toronto, June, 1986.

6. Y. Li and W.M. Wonham, " Controllability and observability in the state-feedback control of discrete-event systems," Systems Control Group Report #8803, Dept. of E. E., Univ. of Toronto, July, 1988. Also, to appear: Proc. 25th IEEE Conference on Decision and Control, December, 1988.

7. J. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1981.

8. P.J. Ramadge and W.M. Wonham, " Modular feedback logic for discrete event systems," *SIAM J. Control and Optimization*, 25(5), 1987, pp. 1202-1218.

9. W.M. Wonham, "A control theory for discrete-event systems," Systems Control Group Report #8714, Dept. of Electrical Eng., Univ. of Toronto, December, 1987.

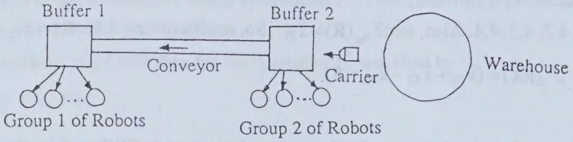
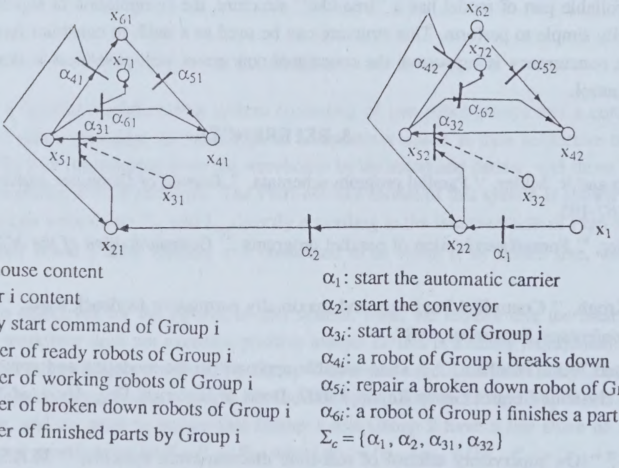

Buffer 1
Buffer 2
Conveyor
Carrier
Warehouse
Group 1 of Robots
Group 2 of Robots

Fig.6.1


x_{61}
x_{71}
α_{41}
α_{51}
α_{61}
α_{31}
x_{51}
x_{41}
x_{31}
x_{21}
α_2
x_{62}
x_{72}
α_{42}
α_{52}
α_{62}
α_{32}
x_{52}
x_{42}
x_{32}
x_{22}
α_1
x_1

x_1: warehouse content

x_{2i}: buffer i content

x_{3i}: binary start command of Group i

x_{4i}: number of ready robots of Group i

x_{5i}: number of working robots of Group i

x_{6i}: number of broken down robots of Group i

x_{7i}: number of finished parts by Group i

α_1: start the automatic carrier

α_2: start the conveyor

α_{3i}: start a robot of Group i

α_{4i}: a robot of Group i breaks down

α_{5i}: repair a broken down robot of Group i

α_{6i}: a robot of Group i finishes a part

$\Sigma_c = \{\alpha_1, \alpha_2, \alpha_{31}, \alpha_{32}\}$

Fig.6.2

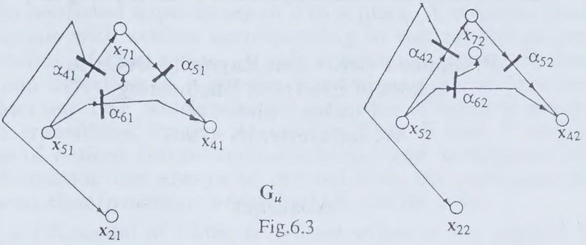
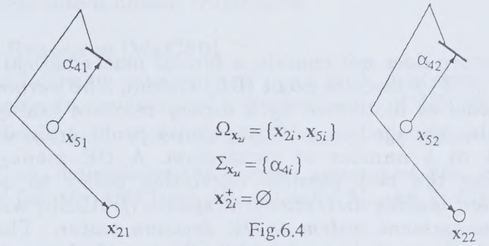

x_{71}
α_{41}
α_{51}
α_{61}
x_{51}
x_{41}
x_{21}
x_{72}
α_{42}
α_{52}
α_{62}
x_{52}
x_{42}
x_{22}
G_u

Fig.6.3

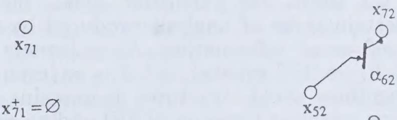

α_{41}
x_{51}
x_{21}
α_{42}
x_{52}
x_{22}

$\Omega_{x_u} = \{x_{2i}, x_{5i}\}$

$\Sigma_{x_u} = \{\alpha_{4i}\}$

$x^+_{2i} = \varnothing$

Fig.6.4


x_{71}
x_{72}
α_{62}
x_{52}

$x^+_{71} = \varnothing$

$\Omega_{x_{72}} = \{x_{72}, x_{52}\}$

$\Sigma_{x_{72}} = \{\alpha_{62}\}$

$x^+_{72} = \varnothing$

Fig.6.5

Modeling of Interacting Continuous Time and Discrete Event Systems : An Example †

Philippos Peleties and Raymond DeCarlo
School of Electrical Engineering
Purdue University
W. Lafayette, IN 47907

Abstract

This paper discusses an example of the management for profit of a fishery resource for the purpose of demonstrating a proposed model for continuous time / discrete event systems (hybrid systems). A Petri Net will subsume the discrete event (management) aspects of the hybrid system while a nonlinear first order state model will subsume the continuous time (growth and harvesting) aspects. Generic event-based interfaces are defined so as to free the model from specific applications.

1. Introduction

To date the literature does not contain a formal mathematical model of a combined continuous time (CT) / discrete event (DE) system. The purpose of this paper is to define of such a model as motivated by a fishery resource example. Specifically a fishery resource is to be managed so as to maximize profit derived from the sale of harvested fish subject to a number of constraints. A DE management system is responsible for choosing the best possible harvesting policy to achieve maximum profit. The term *decision making discrete event system* (DMDES) will be interchangeably used with DE *management system* or DE *decision maker.* The fishery resource has a CT first order non-linear state model while the specific harvesting rate (control) is decided by a DE management system modeled by a Petri Net (PN) [Pet81, Rei85, Mur84]. The actual decision depends on the optimization of a constrained performance index.

The key difficulty in generating a hybrid system (HS) model of such a process is the structure of the interface. Intuitively a management structure makes a decision based on an event in the CT world, the particular "state" the management process currently resides in, and certain types of analysis produced by a processor. This suggests three distinct types of event information, $E_D = \{$events of the management-structure/decision-maker$\}$, $E_C = \{$CT events$\}$ and $E_{PR} = \{$events defined by the output of the processor$\}$. Using these event structures as domains, two interfacing functions are defined, a transition enabling function (TEF) and an actuator function (AF). The TEF, ϕ, maps $E_D \times E_C \times E_{PR}$ to Φ, where $\Phi = B^{n_j}, B = \{0, 1\}$, $n_j \leq n_t$, and n_t is the number of transition in the PN model of the DMDES. The range of this function controls the firing of transitions and thus the evolution of the information flow in the PN. The AF, γ, maps E_D into Γ, where $\Gamma = B^{n_a}$, and n_a is the number of actuators in the CT system (CTS). The use of actuators is convenient but any set of control policies is allowable. Thus events in the DMDES determine the implementation of control policies and/or goals in the CT world.

The motivation behind many of the above ideas, has come from the field of flexible manufacturing systems (FMS). A number of proposed PN-based control schemes appear in literature [KoK84, MaA86, MuK86, VaC85]. In [KoK84, MuK86] a PN-based model of a decision making process is being provided with a set of process I/O functions, δ, ϕ and η which associate places with control and response signals of the continuous time process. In addition a set of system status functions, U and V supervise the execution status of actions and regulate the firing of transitions. Specifically, a command function δ issues place dependent control signals to the continuous time process upon the deposition of a token in the corresponding place. These signals trigger an action in the continuous time world. This function is analogous to our AF which is more general due to the abstraction of events in the decision making process, as opposed to the restricted dependency of δ to a place. A response function ϕ, relates a set of input signals with actions corresponding to control signals issued by δ, indicating the completion of that action. A gate function η is used to open/close transitions, opened upon detection of a specific input signal. These functions play a more restricted role than our TEF which relates a richer set of types of events with the enabling of various transitions. System status functions U and V provide information about the status of actions (execution/completion) and transitions (closed/open). In our case this information can always be derived from the particular state (particular discrete/continuous time/processor events) which the HS is in.

---

† This work supported by the National Science Foundation under Grant No. ECS-8800910.

In [VaC85] a PN-model of FMSs is defined which is less explicit than the previously mentioned models in terms of the interface mechanisms between the CT process and the decision maker. The notions of internal and external events are defined leading to the consistency of E_D and E_C. The actuators of the controlling portion of the system are associated with transitions, being activated when those transitions fire. The various occurrences of events in the continuous time process are being announced to the decision maker through the use of sensors. As will be seen our development abstracts these ideas into a unified framework.

1.1. A Fishery Resource [McC80]

Consider the following scenario : A pond with maximum capacity of M weight units of fish contains an amount x of a single species of fish which is self-reproducing, hence a renewable resource. The rate of increase of the resource, $\frac{dx}{dt}$, in weight units of fish per unit time, depends on the current amount of fish, x, the harvesting rate u in weight units of fish per unit time, the capacity M, and finally a fertility constant K. Thus :

$$\frac{dx}{dt} = Kx(M - x) - u. \tag{1.1}$$

Assuming that $x \geq 0$ and $u \geq 0$, the behavior of the reproduction rate with respect to the amount of fish x and rate of harvesting u is given in Figure 1.1. Figure 1.1 suggests that for a fixed rate of harvesting, there exist two equilibrium points ($\frac{dx}{dt} = 0$) : an unstable point which is closer to zero, and a stable one which is further away. For a specific rate of harvesting, u, if the amount of fish present, x, is to the left of the unstable point, then the resource will gradually be depleted to zero. If on the other hand x is to the right of the same point (right of the stable equilibrium point), the resource will increase (decrease) until the stable equilibrium point is reached. For $u = 0$ the unstable equilibrium point corresponds to $x = 0$ and the stable one to $x = M$.

It can also be seen in Figure 1.1 that for a harvesting rate beyond a given point, the surface representing $\frac{dx}{dt}$ goes below the $u{-}x$ plane, resulting to a negative rate of increase of our resource, implying depletion of that resource to zero.

2. Problem Statement

Our task is to manage the fishery resource in such a way as to both maintain a renewable resource (x to the right of the unstable equilibrium point, for any constant rate of harvesting), yet maximize our profit from the sale of the fish that is being harvested. The assumptions that are made as well as various other considerations are given in the next subsection.

2.1. Assumptions

In the problem of management of a fishery resource, the following assumptions/constraints are made :

1) The fishery is the sole producer and supplier of fish in the market.

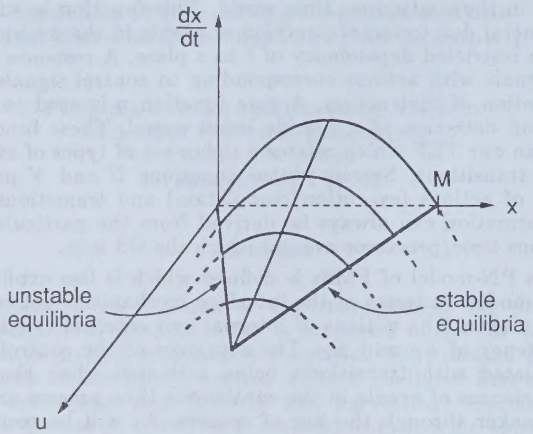


Figure 1.1. Equilibria values of equation 1 governing reproduction rate of fish as a function of (constant) u

2) The demand curve for each week is known from previous market surveys.
3) The amount of fish that is in demand by the market is less than or equal to the amount that is harvested.
4) As a robustness consideration, the harvesting rate cannot deplete the quantity of fish below a certain $x_{\min}$.
5) The harvesting rate is non-negative.

Constraints 3, 4 and 5 have the following mathematical representations respectively :

$$d_0 - ap \leq u \tag{2.1}$$

$$0 \leq x_{\min} \leq x \tag{2.2}$$

$$u \geq 0 \tag{2.3}$$

where p is the price per unit weight of fish sold, d_0 the demand when the price is zero and a a positive constant determining the slope of the demand curve.

2.2. Performance Index

The performance index, J, measures the profitability of the fishery business per unit of time. Specifically

$$J = py - c_0 u \tag{2.4}$$

where y is the amount of fish sold per unit of time and c_0 is an average cost per unit weight of fish. If we further assume that we can harvest/sell as much as our demand requires, then the performance index becomes

$$J = p(d_0 - ap) - c_0 u \tag{2.5}$$

2.3. Harvesting Rate

Our objective is to maximize our profit (maximize equation 2.5), yet maintain the amount of fish within the given bounds. This section determines the optimal harvesting rate.

Figure 2.1 shows the performance index as a function of p and u as well as constraints 3 and 5. By examining the Kuhn-Tucker conditions it is possible to determine

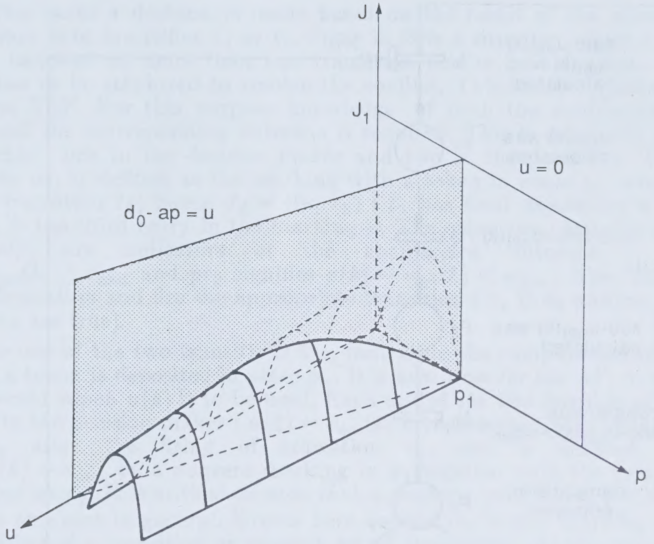


Figure 2.1. Performance index with constraints

the optimal harvesting rate u_{opt} given

$$\underset{u,p}{maximize}\ J = p(d_0 - ap) - c_0 u \tag{2.6}$$

subject to equations 2.1 and 2.3. As we can see in Figure 2.1 the interior of the constraint set corresponds to the portion of the surface representing J which is in front of constraint 3. If constraint 3 is active the performance index is maximum for harvesting rate $u_{opt} = \frac{1}{2}(d_0 - c_0 a)$ and price $p_{opt} = \frac{1}{2}\left(\frac{d_0}{a} + c_0\right)$ which are optimal. The only other case has both constraints active. This corresponds to a unique point of intersection between the two constraints and the surface representing J. That point is p_1, resulting in a harvesting rate of $u_{opt} = 0$. As long as our fish resource is within some given bound, it always pays to use the monetary optimal harvesting rates since they maximize profit.

The above optimal harvesting rate does not take into account the actual amount of fish present in the pond and hence the bound of equation 2.2. For the sake of simplicity, let us assume all fish are harvested at the beginning of the week. If $x - u_{opt} \geq x_{\min}$ then implement $u = u_{opt}$ where in this equation u_{opt} has been multiplied by $\Delta t = 1$ to have consistent units. If $x - u_{opt} < x_{\min}$ then $u = x - x_{\min}$ is implemented.

Given the harvesting process as just described, computation of the amount of fish at the beginning of the next period is necessary. To accomplish this let k denote week k, $x(k)$ denote the amount of fish present and $u(k)$ the amount of fish harvested at week k. Then under the above harvesting scenario

$$x(k+1) = x(k) - u(k) + \int_0^1 Kx(q)(M - x(q))dq \tag{2.7}$$

$$= \frac{M}{1 + \frac{M - (x(k) - u(k))}{x(k) - u(k)} e^{-KM}}$$

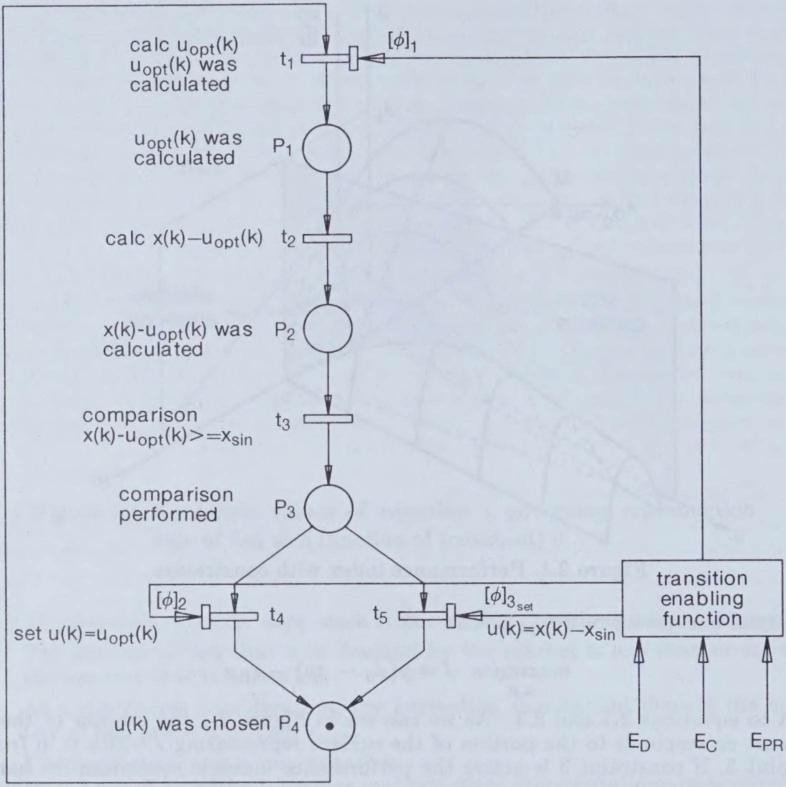

calc $u_{opt}(k)$
$u_{opt}(k)$ was calculated
t_1
$[\phi]_1$
$u_{opt}(k)$ was calculated
P_1
calc $x(k)-u_{opt}(k)$
t_2
$x(k)-u_{opt}(k)$ was calculated
P_2
comparison
$x(k)-u_{opt}(k)>=x_{sin}$
t_3
comparison performed
P_3
transition enabling function
$[\phi]_2$
set $u(k)=u_{opt}(k)$
t_4
t_5
$[\phi]_{3_{set}}$
$u(k)=x(k)-x_{sin}$
u(k) was chosen P_4
E_D
E_C
E_{PR}

Figure 3.1. PN representation of DMDES

Thus it is simple to determine the amount of fish present in the pond in subsequent weeks. In actuality one would sample the pond at the beginning of each week, equation 2.7 being necessary only for simulation purposes.

3. Structure of the Decision Making Process

At this point the discussion has matured enough so that the PN representation of the DMDES can be introduced.

Consider Figure 3.1. This figure represents the flow of decision making. In this figure the first executable decision is the calculation of $u_{opt}(k)$. This calculation takes place in a numerical processor. The execution of the calculation is indicated by the firing of transition t_1 and its completion by a token in place p_1. The calculation will commence only if there is a token in place p_4 and entry one of the TEF, $[\phi]_1$, has a value of one, indicating that the current amount of fish in the pond, $x(k)$, was sampled. The sampling of $x(k)$ constitutes an event in the CT world and it is denoted by c_1.

After $u_{opt}(k)$ is calculated, it is subtracted from the sampled value of $x(k)$ as indicated by the firing of transition t_2. A token in place p_2 indicates the completion of this calculation.

The next computation is a comparison between $x(k) - u_{opt}(k)$ and $x_{\min}$. The firing of transition t_3 and the placement of a token in place p_3 signify the above execution of the calculation and its completion.

At this point a decision is made based on the result of the above comparison. The decision is to fire either t_4 or t_5. Since in PNs a situation where a single marked place p_3 incident on more than one transition results in a conflict, an arbitration method has to be employed to resolve the conflict. This is accomplished through the use of the TEF. For this purpose knowledge of both the completion of the comparison and the corresponding outcome is required. This is presented in the form of three events : one in the decision maker and two in the processor. The first event, denoted as d_3, is defined as the marking with a token in place p_3 resulting from the firing of transition t_3; hence $d_3 = \{(\mu,\ t_3) \mid t_3$ has fired depositing a token in $p_3\}$, where p_3 is the third entry in the marking μ. The other two, denoted as pr_1 and pr_2 respectively, are indicators of the comparison outcome : pr_1 signifies $x(k) - u_{opt}(k) \geq x_{\min}$ and pr_2 signifies $x(k) - u_{opt}(k) < x_{\min}$. The TEF will use the above information and fire the appropriate transition (t_4 if d_3 and pr_1 are true, t_5 if d_3 and pr_2 are true).

After one of the two transitions fire, indicating the computation of the appropriate $u(k)$, a token is deposited in place p_4. It is now time for the AF, γ, to announce to the CT world which $u(k)$ is to be used. Each one of the two possible $u(k)$s constitutes an event in the decision maker : $u(k) = u_{opt}(k)$ is represented by a nonzero marking in place p_4 after the firing of transition t_4, and is denoted by d_1; and $u(k) = x(k) - x_{\min}$ by a nonzero marking in p_4 together with the firing of transition t_5, denoted as d_2. It is critical to note that a nonzero marking in a PN is insufficient to denote an event in general. Events here depend on both a marking and the simultaneous firing of a transition or possibly set of transitions. At this point the marking of the PN is such that the entire decision making process can commence.

Tables 3.1, 3.2, 3.3, 3.4 and 3.5 summarize the various events as well as the actions of the TEF and the AF.

Table 3.1. DM events.

E_D	Representation	Comments
d_1	$\left\{ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^t,\ t_4 \right\}$	u(k) was chosen to be equal to $u_{opt}(k)$
d_2	$\left\{ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^t,\ t_5 \right\}$	u(k) was chosen to be equal to $x(k) - u_{opt}(k)$
d_3	$\left\{ \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^t,\ t_3 \right\}$	comparison between $x(k) - u_{opt}(k)$ and x_{min} was performed

Table 3.2. CT events.

E_C	Comments
c_1	x(k) was sampled
c_2	x(k) waiting to be sampled

Table 3.3. Processor events.

E_{PR}	Comments
pr_1	$x(k) - u_{opt}(k) \geq x_{min}$
pr_2	$x(k) - u_{opt}(k) < x_{min}$

Table 3.4. Actuator function γ.

E_D	γ	Comments
d_1	$\begin{bmatrix} 1 & 0 \end{bmatrix}^t$	Selection of $u(k) = u_{opt}(k)$
d_2	$\begin{bmatrix} 0 & 1 \end{bmatrix}^t$	Selection of $u(k) = x(k) - x_{min}$
*	$\begin{bmatrix} 0 & 0 \end{bmatrix}^t$	Any other case indicates decision in progress

Table 3.5. Transition enabling function ϕ.

E_D	E_C	E_{PR}	ϕ	Comments
$d_1 \vee d_2$	c_1	*	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^t$	$x(k)$ was sampled
d_3	c_2	pr_1	$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^t$	comparison between $x(k) - u_{opt}(k)$ and x_{min} yielded $x(k) - u_{opt}(k) \geq x_{min}$
d_3	c_2	pr_2	$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^t$	comparison between $x(k) - u_{opt}(k)$ and x_{min} yielded $x(k) - u_{opt}(k) < x_{min}$
*	*	*	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^t$	Any other case indicates that ϕ is inactive

4. Definitions of Events and Hybrid System

4.1. CT Events

The notion of "event" in our world is one for which we all have an intuitive feeling: something that happens or the occurrence of a number of actions which satisfy some set of properties. This idea of an event requires observations of the process and the evaluation of the observations to determine if certain conditions are met (e.g. the last five samples of some signal exceed a given threshold). Thus observations correspond to the sampling of the process inputs and responses. In a state model, responses such as state trajectories and input-output trajectories exhibit the modeled behavior of the system for a given set of parameters. Hence events relative to this scenario are the occurrence of samples of the trajectories which have satisfied certain properties. This leads naturally to the following definition of an event in the CT world.

Definition 4.1. An event in the continuous time world is the occurrence of a sample or set of samples that satisfy a particular property or set of properties.

It is instructive to give some examples. Let the notation $c =: a \; R \; b$ mean that some relation $a \; R \; b$ has a Boolean outcome (true/false) which is assigned to the left hand side. The following are common types of events

1) *The sampling of a quantity* : An event defined as $c =: t \in \{i_1, \cdots, i_m\}$, $i_1 < \cdots < i_m$ where t stands for time and i_j an element of an index set $\{i_1, \cdots, i_m\}$. There is no restriction on the index set being finite.
2) *Thresholds* : An event determined by a given quantity being above a given threshold value, i.e. the event $c =: x_i > T_i$, where x_i is a specific scalar quantity (system response) and T_i a given threshold.
3) *Sequence of thresholds* : A sequence of quantities being above a sequence of thresholds, i.e. the event $c =: \bigwedge_{i=1}^{m} x_i > T_i$, where $\wedge$ indicates logical conjunction.
4) *Null event* : The complement of a given set of events within the set of all possible events defined by a particular relation R. For example the event $c =: i_j < t < i_{j+1}$ is the complementary event to (1) above and indicates that a quantity is waiting to be sampled.
5) *Regulation event* : Let $x^k = x(t_k)$ for $k \in \Omega = \{i_1, i_2, \cdots, i_n, \cdots \in Z \mid i_j \leq i_{j+1}$ for any $j\}$; $c =: ||x^k|| < \epsilon$ for every $k \in \{i_j, i_{j+1}, \cdots, i_{j+n}\} \subset \Omega$. Such an event could specify the satisfactory regulation of a given error signal.

Many others are possible. In our fishery example there is an event c_1 given in table 3.2 that the quantity of fish, x, is sampled at the beginning of each week. This exemplifies events of type (1) above where the index set has values {1, 2,..., 52} signifying the beginning of each week in a year. The associated complementary event is the waiting period between successive samples and is denoted by c_2 in table 3.2.

For various sets of CT world events defined by the relations $R_1, \cdots, R_p$ it is clearly possible to build more complex event structures within a particular class defined by R_i or among various classes. For example let $c_1 =: a_1\ R_i\ b_1$ and $c_2 =: a_2\ R_j\ b_2$ then one could take $c_3 =: a_1\ R_i\ b_1 \wedge a_2\ R_j\ b_2$ with no requirement that $i = j$.

4.2. DM Events

The important occurrences in a PN are the firing of transitions and the marking. These facts suggest the following:

Definition 4.2. An event in the DM world is defined as the occurrence of a number of postconditions (or more generally a change in the postconditions) resulting from a simultaneous firing of a set of transitions t_{σ_i}, where σ_i is some index set.

As we can see the above definition suggests that an event in the DM world is identified with a marking-transition pair (μ, t_{σ_i}) as exemplified in table 3.1. The motivation behind this definition is that a marking will indicate that a set of conditions is true and the set of transitions will indicate what course of action led to the condition. In this way, we are able to distinguish between events having the same postconditions arrived at through different courses of action, i.e. arrived at by different paths in the PN. Notice that it is not possible to define an event strictly in terms of a given marking or a specified change in the marking of a set of places. For example a change in the marking of p_4 in Figure 3.1. will not identify the proper control action for the CTS.

It is also possible to define a composition of DM events in the following way :

Definition 4.3. Two DM events are compatible if and only if the postconditions of the first are preconditions of the second: for example let $d_i = \{\mu_i, t_{\sigma_i}\}$, $d_j = \{\mu_j, t_{\sigma_j}\}$, then d_i is compatible with d_j if and only if $\mu_i \longrightarrow \mu_j$ through the firing of the set of transitions t_{σ_j}.

Definition 4.4. The composition of two DM events is defined as the action of creating a macro DM event with the property that its postconditions are those of the second event and its set of transitions is the concatenation of the two sets of transitions, if and only if they are compatible: $\bullet : E_D \times E_D \longrightarrow E_D$ such that $d_i \bullet d_j = (\mu_j, \{t_{\sigma_i}, t_{\sigma_j}\})$ if and only if d_i is compatible with d_j.

4.3. PR Events

From our fishery example it is clear that the thresholding of certain numerical quantities determines various distinct decisions.

Definition 4.5. A processor event is defined as the result of the determination of whether a set of processor quantities satisfies a particular property or set of properties.

Although the above definition is relatively broad, its purpose is to include comparisons between processor quantities having a Boolean result, without excluding any other type of task undertaken by the processor such as numerical computations. In our example, events pr_1 and pr_2 are indicators of the outcome of a comparison between two quantities for greater-than-or-equal-to/less-than respectively.

4.4. HS Model

The above example has introduced a number of ideas. These ideas entail the DMDES and CTS, their underlying models, the two functions TEF and AF which transfer information and control between the two worlds, and finally the events in the discrete, continuous time and processor environments. One more set of notions is necessary before formally defining a HS model. In the previous example the action of the TEF is equivalent to restricting the PN structure to a subnet for each value of ϕ in the discrete set Φ. This implicitly defines a family of nets indexed by Φ. Let us denote this family of nets by PN_Φ. Similarly the DMDES will determine either a control structure change or a system parameter/structure change through the AF, γ. The range of γ is the set (space) Γ which will index the various CTSs as structural changes on the original CTS. Let us denote the family of CTSs indexed by Γ by CTS_Γ. A variable structure controller [DeZ88] or a switched system [LoA87a, LoA87b] are corresponding examples. With these ideas it is now possible to define a HS model.

Formally :

Definition 4.6. A HS model is a 7-tuple

$$(CTS_\Gamma, PN_\Phi, \phi, \gamma, E_D, E_C, E_{PR}) \tag{4.1}$$

where E_D the set of discrete events of the DMDES, E_C the set of CT events of the CTS and E_{PR} the set of processor events, $\phi : E_D \times E_C \times E_{PR} \longrightarrow \Phi$ is the TEF, where $\Phi = B^{n_j}$, $B = \{0, 1\}$, $n_j \leq n_t$, $\gamma : E_D \longrightarrow \Gamma = B^{n_a}$ is the AF, CTS_Γ is a family of CTSs indexed by Φ, PN_Φ is a family of DMDESs indexed by Γ.

The important point here is that the interfaces defined through ϕ and γ are based on generic event structures E_D, E_C, E_{PR}. Thus the model of equation 4.1 is independent of specific applications.

5. Concluding Remarks

Finally, note that time is present in the CTS but not explicit in the PN model. A PN focuses only on the sequential aspect of a system or decision maker. However, a sampler in the CT world will generate a set of samples in a given time period. Hence if one wishes to have the sequential nature of the PN incorporate time, then the event-driven interface structure can be adapted to accomplish this.

6. Acknowledgments

The authors would like to acknowledge Mark Wicks for a number of helpful conversations, Tom Casavant for his questions, and Ruey-wen Liu for first suggesting the use of PN's to us.

7. References

[DeS88] R. A. DeCarlo, S. H. Żak and G. P. Matthews, "Variable Structure Control of Nonlinear Multivariable Systems : A Tutorial," *Proceedings of the IEEE,* Vol. 76, No. 3, March 1988, pp. 212-232.

[KoK84] N. Komoda, K. Kera and T. Kubo, "An Autonomous, Decentralized Control System for Factory Automation," *IEEE Transactions on Computer,* Vol. C-17, 1984, pp 73-83.

[LoA87a] K. A. Loparo, J. T. Aslanis and O. Hajek, "Analysis of Switched Linear Systems in the Plane, Part 1 : Local Behavior of Trajectories and Local Cycle Geometry," *Journal of Optimization Theory and Applications,* Vol. 52, No. 3, March 1987, pp. 365-394.

[LoA87b] K. A. Loparo, J. T. Aslanis and O. Hajek, "Analysis of Switched Linear Systems in the Plane, Part 2 : Global Behavior of Trajectories, Controllability and Attainability," *Journal of Optimization Theory and Applications,* Vol. 52, No. 3, March 1987, pp. 395-427.

[McC80] N. H. McClamroch, *State Models of Dynamic Systems,* Springer-Verlag Inc., New York, NY, 1980, pp. 99-104.

[MaA86] J. Martinez, H. Alla and M. Silva, "Petri Nets for the Specification of FMSs," Modeling and Design of flexible Manufacturing Systems, Andrew Kusiak Elsevier Science Publishers B. V. ed., Amsterdam, 1986, pp. 389-406.

[Mur84] T. Murata, *Petri Nets and their Application : An Introduction,* Management and Office Information Systems, Plenum Publishing Corporation, 1984, pp. 351-368.

[MuK86] T. Murata, N. Komoda, K. Matsumoto and K. Haruna, "A Petri Net-Based Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation," *IEEE Transactions on Industrial Electronics,* Vol. IE-33, No. 1, February 1986, pp. 1-8.

[Pet81] J. L. Peterson, *Petri Net Theory and the Modeling of Systems,* Prentice-Hall, Inc., Engelwood Cliffs, NJ, 1981.

[Rei85] W. Reisig, "Petri Nets : An Introduction," Springer-Verlag, Berlin, 1985.

[VaC85] R. Valette, M. Courvoisier, H. Demmou, J. M. Bigou and C. Desclaux, "Putting Petri Nets to work for Controlling Flexible Manufacturing Systems," *Proceedings of ISCAS,* 1985, pp. 929-932.

BRANCHING TIME TEMPORAL LOGIC FOR DISCRETE EVENT SYSTEM ANALYSIS

K.M. PASSINO and P.J. ANTSAKLIS
Dept. of Electrical and Computer Engineering
University of Notre Dame, Notre Dame IN 46556

Abstract

Given a linear-time temporal logic description of discrete event system behavior, it has been shown that a proof system can be used to prove that certain properties stated in the linear-time temporal language are satisfied. Although such proofs offer an analysis method for discrete event systems, they are often quite difficult to construct, requiring special insights and clever tricks. As an alternative, design specifications can be stated in a branching-time temporal language and a verification algorithm, which replaces the hand-constructed proofs, can be used to test whether the specifications are true or not. In this paper, a certain Petri net is used to model finite state discrete event systems. It is shown that the Petri net model can be used to generate the structure over which the verification algorithm operates. Hence, the algorithm can be used to verify discrete event system properties that can be stated in a branching-time temporal language. The approach to analysis presented in this paper is illustrated by giving three new examples where the Petri nets are specified and the algorithm is used to verify a variety of design specifications.

1.0 Introduction

This paper contains some initial results on the use of a branching-time temporal logic called Computation Tree Logic (CTL) [4,3] for discrete event system analysis in the field of control theory. CTL formulas are used here to state the desired properties or the design specifications that we would like the system to satisfy. A certain Petri net is used to model the underlying discrete event system and to generate a CTL structure which is used by a design specification verification algorithm. Three new examples are given to illustrate the approach to discrete event system analysis. The first example involves a surge tank where the design specifications describe how a liquid level should be regulated. A manufacturing system is used for the second example and a mutual exclusion and priority specification are verified. The third example involves the study of a "self stabilizing distributed system". It is shown that a certain system which was originally thought to be self stabilizing will actually deadlock.

An discrete event system from chemical process control is the "surge tank" [9]. It will be used below as an illustrative example in the explanation of the analysis process for discrete event system studied here. The surge tank has a sensor which can distinguish between five liquid levels (empty, low, normal, high, and full) and a controller which simply opens and closes a fill valve. Unpredictable users operate an empty valve to take liquid from the tank. One design specification for the surge tank system is that the tank never become empty or full if it is initially at a normal liquid level.

First, the discrete event system is modelled with a certain Petri net. The Petri net model defined here allows for control inputs, outputs, and disturbances. There are both control inputs that can cause an immediate change in the plant, and inputs that act to enable and disable certain events. Although the modelling methodology used here is quite flexible, it is limited by a finite state restriction. The Petri net model for the surge tank has one input, the fill valve, one output, the liquid level, and one disturbance, the empty valve. Once the discrete event system has been modelled one must provide the design specifications, i.e., how one would like the system to behave. In this paper the design specifications are stated with CTL formulas. CTL formulas are comprised of (i) atomic propositions such as "the liquid level is low" or "the fill valve is open", (ii) the Boolean connectives (e.g., $\wedge$ and, $\vee$ or, $\neg$ not), and the temporal operators (iii) $\forall s \odot (f_1)$ ($\exists s \odot (f_1)$), intuitively meaning for all (for some) *next* structure state(s) the formula f_1 is true, and (iv) $\forall p[f_1 U f_2]$ ($\exists p[f_1 U f_2]$), intuitively meaning for all (for some) path(s) f_1 is true *until* f_2 becomes true. Suppose that the initial liquid level of the surge tank is normal and the fill valve is open. An example design specification for the surge tank is $\neg\exists s[\mathrm{True} U(\mathrm{empty} \vee \mathrm{full})]$. This intuitively means "it is not the case that there exists a path such that the liquid level will become empty or full". The Petri net model of the discrete event system is used to generate a CTL structure. The design specification verification algorithm from [4] is used here to determine whether a CTL formula is true or not for the CTL structure.

Hence, discrete event system analysis here involves (i) modelling the discrete event system with a certain Petri net, (ii) using the Petri net to generate a CTL structure, (iii) stating the design specifications in the branching time temporal language, and (iv) using a verification algorithm to mechanically test whether the design specifications are true or not. Next, the relationships to relevant research in temporal logic and the theory of Petri nets are given.

Note that, given a linear-time temporal logic description of discrete event system behavior, it has been shown that a proof system can be used to prove that certain properties stated in the linear-time temporal language are satisfied [12,13,18,14]. Although such proofs offer an analysis method for discrete event systems, they are often quite difficult to construct, requiring special insights and clever tricks. In this paper we study the use of a branching-time temporal logic called Computation Tree Logic (CTL) [4,3]. This logic system is similar to the ones found in [1,6]. Design specifications are stated in the CTL language and an algorithm which replaces the hand-constructed proofs is used to test whether the specifications are true or not. A branching-time temporal language is used here since there is a design specification verification algorithm that has complexity linear in both the size of the structure and the length of the specification. In [14] similar efficient decision procedures for simple (low length) formulas in his linear-time temporal language are developed. The verification algorithm for linear-temporal logic in general, however, must have high complexity [4]. The two types of logic must also be compared based on their expressiveness, i.e., what sort of properties can be stated in the temporal language. It is interesting to note that neither linear nor branching-time temporal logics are more expressive than the other; each can express things that the other cannot [11]. For example, linear-time temporal logic cannot express the existence of paths and branching-time cannot express certain "fairness properties". For a complete, up to date discussion of the advantages and disadvantages of linear and branching-time temporal logics see [14,6].

Petri nets appropriate for a discrete event control theoretic setting were first defined in [10]. Analysis results for discrete event systems represented with a sort of Petri net were also developed by Ichikawa and Hiraishi in [7]. The Petri net definition here follows and extends those given in [17,15,16]. It is more expressive than those in [7] and [10] in that it allows for the use of an "inhibitor arc". The expressiveness of the Petri net is, however, limited here by a finite state restriction. The "state equations" described in [17] for Petri nets are used here. These equations are analogous to the state equations used in conventional control theory. Such recursive state equations for discrete event system description have also been given in [8]. The Petri net is used here to generate the CTL structure for a discrete event system. It can be used as an alternative to the "State Machine Language" (SML) described in [2,4] for producing the structure.

The branching-time temporal logic CTL is defined in the next section. The design specification language and verification algorithms are described. In Section 3, the definition of the Petri net used here is given and it is explained how to produce the CTL structure. In Section 4, three examples are described in detail and certain results of this paper are applied to these examples.

2.0 Branching Time Temporal Logic

In this section the design specification language, i.e., the language used to state discrete event system properties is given. The semantics of the language are defined relative to a structure which is generated here with the Petri net defined in the next section. The verification algorithm which operates over the structure is described. The results of this section follow those of [4] with some notational differences.

2.1 The Design Specification Language

The specification language is a propositional branching-time temporal logic called "Computation Tree Logic" (CTL) reported in [4,3]. The language syntax is given first. Let **N** denote the set of natural numbers. Let $\mathbf{A_p}$ denote the underlying finite set of *atomic propositions*. The *formulas* of the CTL are defined inductively.

(i) Every atomic proposition $p \in \mathbf{A_p}$ is a CTL formula.

(ii) If f_1 and f_2 are CTL formulas, then so are $\neg f_1$, $f_1 \wedge f_2$, $\forall s \odot (f_1)$, $\exists s \odot (f_1)$, $\forall p[f_1 \mathbf{U} f_2]$, and $\exists p[f_1 \mathbf{U} f_2]$.

(iii) Nothing else is a CTL formula unless it is obtained by finitely many applications of (i) and (ii) above.

The symbols $\wedge$ and $\neg$ mean "and" and "not" respectively. The parentheses "(" and ")" are used as needed for forming formulas. The *next time* operator is $\odot$. The formula $\forall s \odot (f_1)$ ($\exists s \odot (f_1)$) intuitively means that f_1 holds in every (in some) immediate successor structure state of the current structure state. The *until* operator is **U**. The formula $\forall p[f_1 \mathbf{U} f_2]$ ($\exists p[f_1 \mathbf{U} f_2]$) intuitively means that for every path of structure states (for some path of structure states), there exists an initial prefix of the path such that f_2 holds at the last structure state of the prefix and f_1 holds at all other structure states along the prefix.

The semantics of CTL formulas are defined relative to a labeled state-transition graph (or Kripke structure). A CTL structure is a triple $\mathbf{M}=(\mathbf{S},\mathbf{R},\mathbf{P_c})$ where

(i) **S** is a nonempty finite set of *structure states*.

(ii) **R** is a binary relation on **S** ($\mathbf{S}x\mathbf{S} \supseteq \mathbf{R}$) which gives the possible transitions between states and must be total, i.e. $\forall s \in \mathbf{S}\ \exists s' \in \mathbf{S}$ such that $(s,s') \in \mathbf{R}$.

(iii) $\mathbf{P_c}:\mathbf{S} \rightarrow 2^{\mathbf{A_p}}$ assigns to each structure state the set of atomic propositions true in that structure state.

A *path* is an infinite sequence of structure states denoted by $<s_0, s_1, s_2, \ldots>$ rooted at s_0 such that $\forall i \in \mathbf{N}$, $(s_i,s_{i+1}) \in \mathbf{R}$. The standard notation for indicating truth in a structure is used: $\mathbf{M},s_0 \models f$ means that formula f holds at structure state s_0 in structure **M**. When the structure **M** is understood $s_0 \models f$ is used. Assume that f, f_1, and f_2 are CTL formulas. The relation $\models$ is defined inductively as follows.

(i) $s_0 \models a$ iff $a \in \mathbf{P}_c(s_0)$.

(ii) $s_0 \models \neg f$ iff it is not the case that $s_0 \models f$.

(iii) $s_0 \models f_1 \wedge f_2$ iff $s_0 \models f_1$ and $s_0 \models f_2$.

(iv) $s_0 \models \forall s \odot (f)$ iff for all states s such that $(s_0,s) \in \mathbf{R}$, $s \models f$.

(v) $s_0 \models \exists s \odot (f)$ iff for some state s such that $(s_0,s) \in \mathbf{R}$, $s \models f$.

(vi) $s_0 \models \forall p[f_1 \mathbf{U} f_2]$ iff for all paths $p=<s_0, s_1, s_2, \ldots>$, $\exists i \in \mathbf{N}$ where $i \geq 0$ and $s_i \models f_2$, and $\forall j \in \mathbf{N}$ where $0 \leq j < i$, $s_j \models f_1$.

(vii) $s_0 \models \exists p[f_1 \mathbf{U} f_2]$ iff for some path $p=<s_0, s_1, s_2, \ldots>$, $\exists i \in \mathbf{N}$ where $i \geq 0$ and $s_i \models f_2$, and $\forall j \in \mathbf{N}$ where $0 \leq j < i$, $s_j \models f_1$.

The atomic proposition "True" is always true, i.e. for all $s \in \mathbf{S}$, $\mathrm{True} \in \mathbf{P}_c(s)$. The following abbreviations are used in writing CTL formulas.

(i) $f_1 \vee f_2 \equiv \neg(\neg f_1 \wedge \neg f_2)$ for logical disjunction.

(ii) $f_1 \rightarrow f_2 \equiv \neg f_1 \vee f_2$ for logical implication.

(iii) $f_1 \leftrightarrow f_2 \equiv (f_1 \rightarrow f_2) \wedge (f_2 \rightarrow f_1)$ for logical equivalence.

(iv) $\forall p \Diamond (f) \equiv \forall p[\mathrm{True} \mathbf{U} f]$ intuitively means that f holds in the future along every path p from s_0, i.e. f is *inevitable*.

(v) $\exists p \Diamond (f) \equiv \exists p[\mathrm{True} \mathbf{U} f]$ intuitively means that there is some path p from s_0 that leads to a structure state at which f holds, i.e., f *potentially* holds.

(vi) $\forall p \boxdot (f) \equiv \neg \exists p \Diamond (\neg f)$ intuitively means that f holds at every structure state on every path p from s_0, i.e., f holds *globally*.

(vii) $\exists p \boxdot (f) \equiv \neg \forall p \Diamond (\neg f)$ intuitively means that there is some path p on which f holds at every structure state.

This completes the definition of the design specification language. Once a design specification is stated in the branching-time temporal language it is tested whether it is true or not with a verification algorithm that is described next.

2.2 The Design Specification Verification Algorithm

In this section a brief outline of the design specification verification algorithm is presented. More details are given in [4]. The verification algorithm is an effective procedure for determining whether a CTL formula is true or not. It is a "model checker" since it determines if the structure **M** is a model for the CTL formula being tested. Assume that we wish to determine whether formula f_0 is true or not in the finite structure **M**. Roughly speaking, the verification algorithm iteratively labels the states of the structure **M** with successively longer subformulas of f_0 (and negations of subformulas of f_0) until structure states are labelled with the formula f_0 or $\neg f_0$. With this, the algorithm can determine whether the formula f_0 is true or not. A more detailed description follows.

Let $\mathbf{sub}(f_0)$ denote the nonempty finite set of *subformulas* of CTL formula f_0. Note that all formulas $f \in \mathbf{sub}(f_0)$ are also CTL formulas. The definition of subformulas proceeds by induction on f_0.

(i) If $f=f_0$, then $f \in \mathbf{sub}(f_0)$.

(ii) If $f \in \mathbf{sub}(f_0)$ and $f=\neg f_1$, $f=\forall s \odot (f_1)$, or $f=\exists s \odot (f_1)$, then $f_1 \in \mathbf{sub}(f_0)$.

(iii) If $f \in \mathbf{sub}(f_0)$ and $f=f_1 \wedge f_2$, $f=\forall p[f_1 \mathbf{U} f_2]$, or $f=\exists p[f_1 \mathbf{U} f_2]$, then $f_1, f_2 \in \mathbf{sub}(f_0)$.

(iv) Nothing else is a subformula of f_0.

The nonempty finite set of *positive subformulas* of f_0 is given by $\mathbf{sub}^+(f_0)=\{f' \mid$ for all $f \in \mathbf{sub}(f_0)$, if $f=\neg f_1$, then $f'=f_1$ else $f'=f\}$. Next, we define the *length* of a formula f, denoted by $\rho(f)$. The definition proceeds by induction on f.

(i) $\rho(f)=1$ if $f=p$ where $p \in \mathbf{A}_p$, the set of atomic propositions,

(ii) $\rho(f)=\rho(\neg f)$,

(iii) $\rho(\forall s \odot (f_1))=\rho(\exists s \odot (f_1))=\rho(f_1)+1$,

(iv) $\rho(f_1 \wedge f_2)=\rho(\forall p[f_1 \mathbf{U} f_2])=\rho(\exists p[f_1 \mathbf{U} f_2])=\max\{\rho(f_1),\rho(f_2)\}+1$.

The notion of length of a formula is similar to "rank" [9] in that it quantifies the complexity of the formula.

Let **label**(s) denote the set of formulas that the algorithm has labeled structure state $s \in \mathbf{S}$ with. Beginning with i=1, on pass k the verification algorithm labels each structure state $s \in \mathbf{S}$ with the set of formulas f in $\mathbf{sub}^+(f_0)$ (or their negations) of length $\rho(f)=k$ so that $f \in \mathbf{label}(s)$ iff $\mathbf{M},s \models f$ and $\neg f \in \mathbf{label}(s)$ iff $\mathbf{M},s \models \neg f$. The algorithm makes n passes where $n=\mathrm{length}(f_0)$. Hence, at termination $f_0 \in \mathbf{label}(s)$ iff $\mathbf{M},s \models f_0$ and $\neg f_0 \in \mathbf{label}(s)$ iff $\mathbf{M},s \models \neg f_0$. The actual algorithm that was used here is given in [4]. It runs in time $O(\rho(f_0)(|\mathbf{S}|+|\mathbf{R}|)^2)$. There is another algorithm for solving this problem that runs in time $O(\rho(f_0)(|\mathbf{S}|+|\mathbf{R}|))$ [3].

3.0 A Petri Net Representation for Discrete Event Systems

In this section a Petri net modelling methodology is developed that is used to produce the structure **M**. A *multiset* (bag) is a collection of objects over some domain **X**, but unlike standard definitions of a set, multisets allow multiple occurrances of elements [17]. Let **B** be a multiset, then $\#(x,\mathbf{B})$ represents the number of occurrences of element x in multiset **B**. The set $\mathbf{X}^{\infty}$ is the set of all multisets over a domain **X**. If $x,y\in \mathbf{N}^n$, $x=[x_1,x_2, \ldots ,x_n]^t$, and $y=[y_1,y_2, \ldots ,y_n]^t$ (t indicates transpose) then the statement $x\geq y$ is true iff $x_i\geq y_i$ $1\leq i\leq n$. Similarly for $>,<$, and $\leq$. Let Ω_n denote the set containing the column vetors of an n-dimensional identity matrix and an n-dimensional vector of zeros. Let Ø denote the null set.

The Petri Net structure is described by $\mathbf{P_S}=(\mathbf{P},\mathbf{U},\mathbf{Y},\mathbf{T},\mathbf{I_D},\mathbf{I_N},\mathbf{O_D},\Gamma,\Psi)$ where:

(i) $\mathbf{P}=\{p_1,p_2, \ldots ,p_n\}$ is a non-empty finite set of $n=|\mathbf{P}|$ (*state*) *places* represented graphically with circles (○).

(ii) $\mathbf{U}=\{u_1,u_2, \ldots , u_{n_c}\}$ is a finite set of $n_c=|\mathbf{U}|$ *control places* represented graphically with circles as in (i).

(iii) $\mathbf{Y}=\{y_1,y_2, \ldots ,y_{n_y}\}$ is a finite set of $n_y=|\mathbf{Y}|$ *measurement places*, represented graphically with circles as in (i).

(iv) $\mathbf{T}=\{t_1,t_2, \ldots ,t_m\}$ is a non-empty finite set of $m=|\mathbf{T}|$ *transitions* represented graphically by line segments (|).

Note that $\mathbf{P}\cap\mathbf{U}\cap\mathbf{Y}\cap\mathbf{T}=\emptyset$. Let $\Pi=\mathbf{P}\cup\mathbf{U}$ and $n_\pi=|\Pi|$.

(v) $\mathbf{I_D}:\mathbf{T}\rightarrow\Pi^{\infty}$ is a mapping from transitions to the set of all multisets over Π. It is represented graphically by *directed arcs* (⟶) pointing from each *input place of* t_j, $\pi\in \mathbf{I_D}(t_j)$, to t_j. If for some $\pi\in\Pi$, $\#(\pi,\mathbf{I_D}(t_j))=k>1$, then the mapping can be represented graphically by a directed arc with multiplicity k ($\xrightarrow{k}$). Note that k is finite.

(vi) $\mathbf{I_N}:\mathbf{T}\rightarrow\Pi^{\infty}$ is a mapping from transitions to the set of all multisets over Π. It is represented graphically by *not arcs* (inhibitor arcs) (—○) pointing from each input place of t_j, $\pi\in \mathbf{I_N}(t_j)$, to t_j. If for some $\pi\in\Pi$, $\#(\pi,\mathbf{I_N}(t_j))=k>1$, then the mapping can be represented by a not arc with multiplicity k ($\overset{k}{\text{—○}}$). Note that k is finite.

(vii) $\mathbf{O_D}:\mathbf{T}\rightarrow\mathbf{P}^{\infty}$ is a mapping from transitions to the set of all multisets over **P**. It is represented graphically by directed arcs pointing from the transition $t_j\in \mathbf{T}$ to each *output place of* t_j, $p_i\in \mathbf{O_D}(t_j)$. A directed arc with multiplicity k can be used in a manner similar to (v), except $k=\#(p_i,\mathbf{O_D}(t_j))$.

(viii) $\Gamma:\mathbf{T}\rightarrow\mathbf{U}$ is a mapping from transitions to control places. It is represented graphically by *control arcs* (—▪) pointing from each $u_i\in\Gamma(t_j)$ to t_j. It is required that $\#(u_i,\Gamma(t_j))\leq 1$ for all $t_j\in \mathbf{T}$ and $u_i\in \mathbf{U}$.

(ix) $\Psi:\mathbf{P}\rightarrow\mathbf{Y}$ is a mapping from places to measurement places. It is represented graphically with a *connection arc* (▷—◁) from $p_i\in \mathbf{P}$ to $y_j\in\Psi(p_i)$. In this paper we require that for all $p_i\in \mathbf{P}$ there exist a unique $y_j\in\Psi(p_i)$.

The set $\mathbf{T_c}=\{t_j|$ there exists some u_i, $u_i\in\Gamma(t_j)\}$ and $\mathbf{T_{uc}}=\mathbf{T}-\mathbf{T_c}$. For convenience, a *two-way directed arc* (◀—▶) is used to indicate a *self loop*, i.e. for $p_i\in \mathbf{P}$, $p_i\in \mathbf{O_D}(t_j)$ and $p_i\in \mathbf{I_D}(t_j)$. A *directed not arc* (◀—○) is used to indicate the connections $p_i\in \mathbf{O_D}(t_j)$ and $p_i\in \mathbf{I_N}(t_j)$, where $p_i\in \mathbf{P}$. Also, every arc (except the connection arc) defined above has a transition on one end and a place on the other, and no transition or place exists without being connected to an arc.

A complete description, which also includes the *execution* characteristics of the Petri Net, is given by $\mathbf{P_N}=(\mathbf{P_S},\mathbf{T_s}\mathbf{X_p},\mathbf{X_{po}},\mathbf{U_p},\mathbf{U_a},\mathbf{Y_p},\mathbf{Y_a},\mathbf{E_r},\Phi)$ where:

(i) $\mathbf{P_S}$ is described above.

(ii) $\mathbf{T_s}$ is the *time index set* which is specified according to the modelling problem at hand. Often $\mathbf{T_s}=\mathbf{N}$ and the initial time is 0, and each successive natural number represents an arbitrary length time step. Alternatively, the natural numbers represent fixed length time units such as seconds.

(iii) $\mathbf{X_p}:\mathbf{P}\mathrm{x}\mathbf{T_s}\rightarrow\mathbf{N}$ is the *marking function,* a mapping from (state) places and time steps into nonnegative integers representing the *marking* of the place. The n-dimensional column vector $x_p(k)=[X_p(p_1,k),X_p(p_2,k), \ldots ,X_p(p_n,k)]^t$ is used to denote the *state* of the Petri net. The state is represented graphically by *tokens* (●) that are put inside places (e.g. $X_p(p_i,k)=2$ is represented as P_i (●●)). The complete set of "reachable states" [17] will be denoted by $\mathbf{X_p}$. It is required that $X_p(p_i,k)$ be finite for all $p_i\in \mathbf{P}$ and $k\in \mathbf{T_s}$.

(iv) $\mathbf{X_{po}}$ is a non-empty finite set of *initial conditions* $x_p(0)$ for the state of the Petri Net; $\mathbf{N}^n\supseteq\mathbf{X_{po}}$.

(v) $\mathbf{U_p}:\mathbf{U}\mathrm{x}\mathbf{T_s}\rightarrow\mathbf{N}$ is a mapping from control places and time steps into nonnegative integers representing the marking of the control place. It is represented graphically with tokens as in (iii). The n_c-dimensional column vector $u_p(k)=[U_p(u_1,k), U_p(u_2,k), \ldots ,U_p(u_{n_c},k)]^t$ is used to denote the *control input* to the Petri net. It is required that $u_p(k)\in \Omega_{n_c}$ for all $k\in \mathbf{T}_s$.

(vi) $\mathbf{U}_a$ is a non-empty finite set of admissible control inputs $u_p(k)$ for the Petri Net.

(vii) $Y_p:\mathbf{Y}\mathrm{x}\mathbf{T}_s\rightarrow \mathbf{N}$ is a mapping from measurement places and time steps to a nonnegative integer representing the marking of the measurement place. Let $Y_p(y_j,k)=X_p(p_i,k)$ for all $k\in \mathbf{T}_s$ where $y_j\in \Psi(p_i)$. The $n_y(n)$-dimensional column vector $y_p(k)=[Y_p(y_1,k), Y_p(y_2,k), \ldots ,Y_p(y_{n_y},k)]^t$ is used to denote the *output* of the Petri net.

(viii) $\mathbf{Y}_a$ is a finite set of admissible outputs $y_p(k)$ for the Petri Net, which, for this paper, is $\mathbf{X}_p$.

(ix) $\mathbf{E}_r:\mathbf{N}^{n_\pi}\mathrm{x}\mathbf{T}_s\rightarrow 2^{\mathbf{T}}$ is the Petri net *enable rule*, a mapping from an n_π-dimensional column vector of nonnegative integers representing the marking of the state and control places and time steps into subsets of transitions that are said to be *enabled* at step k. The notation $t_j\in \mathbf{E}_r(k)$ is used to indicate that t_j is enabled at step k. A transition t_j can *fire* whenever it is enabled. Tokens are redistributed in the Petri net when a transition fires according to the next state function described below. Following the definition in [17] it is assumed that no two transitions will fire at exactly the same time. The set of enabled transitions is formed at time k, then one is chosen to be fired. Also, the transitions fire only at times $k\in \mathbf{T}_s$.

(x) $\Phi:\mathbf{T}\mathrm{x}\mathbf{N}^n\mathrm{x}\mathbf{T}_s\rightarrow \mathbf{N}^n$ is the Petri net *next state function (firing rule)*, a mapping from transitions, an n-dimensional column vector of nonnegative integers representing $x_p(k)$, and time steps into an n-dimensional column vector of nonnegative integers representing the next state $x_p(k+1)$. The next state function is defined at step k iff $t_j\in \mathbf{E}_r(k)$. Suppose that a transition fires at time k (or if k=0 then none has fired), then the transition which fires next is chosen in the following manner. If $t_j\in \mathbf{E}_r(k)$ and $t_j\in \mathbf{T}_c$ then the transition t_j is fired at time $k+\Delta t$ where $\Delta t\rightarrow 0$, i.e., immediately after the last transition firing. All enabled transitions $t_j\in \mathbf{T}_c$ are fired, then the enabled transitions $t_j\in \mathbf{T}_{uc}$ are allowed to fire asynchronously. This will allow the plant to respond immediately to an input.

A candidate for the enable rule is given by $\mathbf{E}'_r=\{t_j|X_p(p_i,k)\geq\#(p_i,\mathbf{I}_D(t_j)), X_p(p_i,k)<\#(p_i,\mathbf{I}_N(t_j)), U_p(u_i,k)\geq\#(u_i,\Gamma(t_j)), U_p(u_i,k)\geq\#(u_i,\mathbf{I}_D(t_j)), U_p(u_i,k)<\#(u_i,\mathbf{I}_N(t_j))$, for all $p_i\in \mathbf{P}, u_i\in \mathbf{U}\}$. The enable rule is chosen based on the specific modelling task. This enable rule is used for all the examples in this paper. Let $A^-=[a^-_{ij}]$, where $a^-_{ij}= \#(p_i,\mathbf{I}_D(t_j))$ and $A^+=[a^+_{ij}]$, where $a^+_{ij}=\#(p_i,\mathbf{O}_D(t_j))$. Let $A=A^+-A^-$. Let $e_j(k)=[0\ 0 \ldots 1 \ldots 0\ 0]^t$ where the 1 is in the jth position $1\leq j\leq m$. Let $C=[c_{ij}]$ with $c_{ij}=1$ if $y_i\in \Psi(p_j)$ and $c_{ij}=0$ if $y_i\notin \Psi(p_j)$. Let $\Phi=[\phi_1,\phi_2, \ldots ,\phi_n]^t$. Using $\mathbf{E}'_r$ as the enable rule, an example next state function is $\phi_i(t_j,X_p(p_i,k),k) = X_p(p_i,k+1)=X_p(p_i,k) - \#(p_i,\mathbf{I}_D(t_j)) + \#(p_i,\mathbf{O}_D(t_j))$ for all i, $1\leq i\leq n$. Hence, for $t_j\in \mathbf{E}'_r(k)$ firing we have,

$$x_p(k+1)=x_p(k) + Ae_j(k)$$
$$y_p(k)=Cx_p(k)$$

which are similar to the *state equations* described in [17]. For this next state function tokens are not removed from any control or disturbance place if t_j fires, but they must be present to enable t_j. Likewise, tokens are not removed from places connected to transitions via a not arc. The connection arcs between any place p_i and a measurement place y_j indicate that any tokens added or subtracted from p_i are also added and subtracted from y_j, i.e. with the connection arc they are essentially duplicate places.

It must be emphasized that the Petri net defined above is a flexible modelling tool since we do not confine ourselves to the standard definitions of the arcs, enable rules, and firing rules. They can be chosen according to the modelling task at hand. For instance, in the third example on self stabilizing distributed systems it is convenient to define an arc that connects transitions $t_j\in \mathbf{T}$ to places $p_i\in \mathbf{P}$ which if t_j is enabled at step k and is fired then $X_p(p_i,k+1)=1$. Special symbols can be used for the graphical representation of such arcs. For the arc just defined we use the symbol —●▶ .

This completes the definition of the Petri net. Although it is not necessary, a controller can be used for the Petri net. One possible controller for the Petri net is given by $G_c:\mathbf{X}_p\rightarrow \mathbf{U}_a$, a mapping of states of the Petri net into admissible inputs to the Petri net. The initial controller output is the appropriate element of $\mathbf{U}_a$ depending on the initial state of the plant. Note that if a transition in the plant fires, changing $x_{pi}\in \mathbf{X}_p$ to $x_{pi'}\in \mathbf{X}_p$ the control input to the plant is updated instantaneously from $G_c(x_{pi})$ to $G_c(x_{pi'})$.

We only briefly outline how the structure **M** can be formed using the Petri net. Clearly, regardless of whether a controller is used or not, the state equations above provide for the specification of the structure **M**. The structure states **S** are the states of the Petri net and the firing of various transitions specifies the relation **R**. The function $\mathbf{P}_c$ depends on what is to be verified. For instance, if the input-output properties of the plant are of interest then the atomic propositions may be elements of the sets $\mathbf{U}_a$ and $\mathbf{Y}_a$. The CTL structure also depends on the initial state. If there is more than one initial state then the CTL structure may have to be produced for each initial state (depending on if the initial condition is in the CTL structure produced for another initial condition). With **M**, and the design specifications stated in the branching-time temporal language, the verification algorithm is used to prove whether the specifications are true or not.

Several examples of the modelling methodology, design specifications, and their verification are given in the next section.

4.0 Examples

In this section we briefly describe how the above results are applied to three examples. For the first example, consider the surge tank taken from [9] shown in Figure 4.1 with a sensor distinguishing among five liquid levels (empty, low, normal, high, and full) and with the controller simply opening and closing a fill valve. Unpredictable users operate an empty valve to drain the tank. The objective is to verify that a given controller will properly regulate the liquid level in the tank.

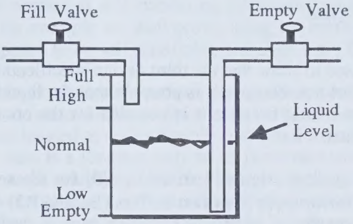


Figure 4.1 Surge Tank

The liquid level "empty" indicates that the tank is absolutely empty, hence if the fill valve is opened the tank will immediately become "low". The liquid level of "low" means that the liquid height is between empty and "normal". "High" is between normal and "full", which indicates that the tank (which is sealed at the top) is completely full. The Petri net model of the surge tank is given in Figure 4.2 below.

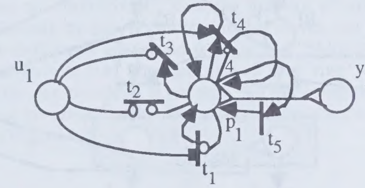


Figure 4.2 Petri Net Model of the Surge Tank

The place p_1 is used to represent the state of the system which is the liquid height. The interpretation of time is that each successive $k \in \mathbf{T}_s$ represents one second. The cases where $x_p(k)=X_p(p_1,k)=i$, with $i=0,1,2,3,4$ represent the five liquid levels in the tank (empty, low, normal, high, and full respectively). Denote these states with atomic propositions x_{pi}, $i=0,1,2,3,4$ respectively. The connection arc joins the measurement place y_1 to the state place p_1, hence the measured output is $y_p(k)=x_p(k)$, the state of the system. The input place u_1 represents the fill valve, and $u_p(k)=1$ indicates that the fill valve is on while $u_p(k)=0$ indicates that it is off. The set of transitions $T=\{t_1,t_2,t_3,t_4,t_5\}$. Each of these transitions specifies certain rules about the behavior of the plant. Transition (i) t_1 indicates that if the liquid level is empty ($x_p(k)=0$) and the fill valve is on, then the liquid level will immediately become low ($x_p(k)=1$), (ii) t_2 indicates that if the liquid level is empty and the fill valve is closed then the liquid level will stay empty, (iii) t_3 models the case where if there is a liquid level above empty and the fill valve is off then the liquid level may drop one level depending on the empty valve, (iv) t_4 indicates that if the liquid level is less than full ($x_p(k)<4$) and it is greater than or equal to the empty level and the fill valve is on, then the liquid level may raise depending on the empty valve, and (v) t_5 models the case where if the liquid level is above empty it may be that the next state is exactly the same as the previous one.

Next, the controller for the surge tank is specified. For the surge tank the set of reachable states is $X_p=\{0,1,2,3,4\}$ and the set of admissible inputs is $U_a=\{0,1\}$. The controller for the surge tank, G_c, is given by: (i) $G_c(0)=G_c(1)=G_c(2)=1$ and (ii) $G_c(3)=G_c(4)=0$. Using the controller and the Petri net model of the surge tank, the CTL structures appropriate for each of the following design specifications were generated for the verification algorithm.

The first two design specifications involve verifying that if the liquid level is initially normal ($x_p(0)=2$) and the fill valve is open then (i) it will not be the case that the tank will become full or empty, and (ii) it is inevitable that the level will always be either normal or full. Stated formally these two design specifications are:

(i) $\forall p \boxdot (\neg(x_{p0} \vee x_{p4}))$, and

(ii) $\forall p \Diamond (\forall p \boxdot (x_{p2} \vee x_{p3}))$

respectively. These two specifications were easily verified using the algorithm described in Section 2.2. The third and fourth design specifications consider the case when the tank is initially empty ($x_p(0)=0$) and the fill valve is initially off. The two specifications are:

(iii) $\forall p \Diamond (\exists p \boxdot (x_{p2} \vee x_{p3}))$, and

(iv) $\exists p \Diamond (\exists p \boxdot (x_{p2} \vee x_{p3}))$.

The verification algorithm was used to show that the third design specification is false and the fourth one is true. The third specification is not true because it is possible that the liquid level end up being forever at the low level and the fourth one is true because it is possible for the controller to raise the level if the empty valve is off sufficiently often.

We use the manufacturing system originally studied in [9] for the second example. We have one machine which must process different parts from two buffers (B1 and B2) each supplied by two different producers (P1 and P2). Upon completion of processing, each type of part is put in its respective output bin. A controller must ensure mutual exclusion in the machine, i.e., only one part of one type is processed at a time. Also, the controller must also ensure that producer P1 gets priority in the use of the machine. To perform its task, the controller can sense whether there is a part in a given buffer and can also determine the type of part in the machine. The controller acts to allow the different parts to enter the machine. The Petri net model of the machine is given in Figure 4.3 below.

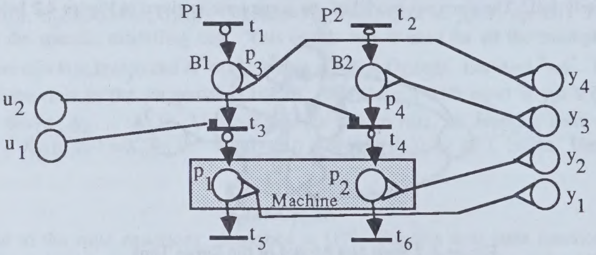


Figure 4.3 Petri Net Model of a Manufacturing System

The producer P1 (P2) is represented with the transition t_1 (t_2). The buffer B1 (B2) is represented with the place p_3 (p_4) and measurement place y_4 (y_3) is used to indicate if a part is in buffer B1 (B2). The place p_1 (p_2) and measurement place y_1 (y_2) are used to represent whether or not a P1 (P2) type part is in the machine. The firing of t_3 (t_4) indicates that parts from B1 (B2) are put in the machine and the firing of t_5 (t_6) indicate that parts are removed from the machines and put into their respective output bins.

The interpretation of time is that each $k \in \mathbf{T}_s$ represents an arbitrary length time step. The state $x_p(k)=[X_p(p_1,k)\ X_p(p_2,k)\ X_p(p_3,k)\ X_p(p_4,k)]^t$. Hence the state $x_p(k)=[1\ 0\ 0\ 1]^t$ means that there is a P1 part in the machine, there is not a P2 part in the machine, there is not a P1 part in buffer B1, and there is a P2 part in buffer B2. We shall use the notation "1001" to represent this state and a similar notation for other states. The input $u_p(k)=[U_p(u_1,k)\ U_p(u_2,k)]^t=[1\ 0]^t$ indicates that a P1 part is to be allowed to be put into the machine. The notation "10" will be used to denote this input and others will be denoted similarly. The set of admissible inputs is contains 10,01, and 00. The controller for the manufacturing system G_c is given by: (i) $G_c(x)=10$ if $x=0010$ or 0011, (ii) $G_c(x)=01$ if $x=0001$, and (iii) $G_c(x)=00$ for all other reachable states. The initial conditions are that the state of the plant is 0000 and the controller output is 00. The CTL structure was generated using the Petri net model of the manufacturing system and the following specifications were considered:

(i) $\forall p \boxdot (\neg(1100 \vee 1101 \vee 1110 \vee 1111))$

(ii) $\forall p \boxdot (0011 \rightarrow \forall s \odot (1001 \vee 1011))$

The first specification is a statement of mutual exclusion. It was verified first. The second specification says that producer P1 should have priority over producer P2. In particular, it says that if it is ever the case where there are parts in both of the buffers then the P1 part should be allowed to enter the machine. This CTL formula was also verified using the verification algorithm. This manufacturing example is similar to the "Two Class Parts Processing" example in [18]. There, however, Thistle and Wonham allow an arbitrary finite number of parts of one type to enter the machine. This forces their controller to have an infinite number of states, so the verification algorithm cannot be applied to their example. There are, however, many practical problems that are finite state.

The third example is taken from Dijkstra's paper "Self Stabilizing Systems in Spite of Distributed Control" where he considers the existence of self stabilizing distributed systems and provides three systems that are supposed to possess this property [5]. Thistle has used a certain linear temporal logic proof system to prove that one of Dijkstra's systems is self stabilizing (although in a different manner than originally thought by Dijkstra) [18]. In this example we shall prove, using the verification algorithm, that another of Dijkstra's machines that is supposed to be self stabilizing is in fact not. The problem formulation follows the one in [18].

The sort of distributed systems considered are those that can be modelled as a sparsely connected finite graph having at each node a finite state machine. Each of these machines can communicate only with its "neighbors", i.e., those machines located at nodes directly connected to its own. If a machine makes a state transition, or "move", its new state is a function only of its previous state and the states of its neighbors. Also, the moves of a machine can be enabled or disabled only on the basis of the states of the machine and its neighbors: in order for a move to occur, a corresponding Boolean condition on the states of the machine and its neighbors must be satisfied. These conditions are called "privileges", and when satisfied, a privilege is said to be "present". A "demon" selects at random one of the privileges that is present and the corresponding move occurs; consequently asynchronous operation of the machine is modelled.

Dijkstra considers a distributed system to be "self stabilizing" if for all initial states it eventually reaches a "legitimate" state. The set of legitimate states can be defined to be any set having the following properties: (i) in every legitimate state, at least one privilege is present; (ii) from any legitimate state, all enabled moves bring the system into another legitimate state; (iii) each privilege is present in at least one legitimate state; and (iv) from any legitimate state it is possible to reach any other legitimate state (as long as the demon happens to select privileges appropriately). The concern is with whether self stabilizing distributed systems exist that are "non-trivial", meaning that not all system states are legitimate states. Dijkstra proposes three solutions to this problem, but he leaves the proof that the systems are self stabilizing to the reader. We shall study his second solution, the "Solution with Four-State Machines".

Dijkstra confines the his study to N+1 machines placed in a "ring", and considers legitimate states to be the states in which exactly one privilege is present. In his solution using four-state machines each machine in the ring has four possible states. We consider the simple case where there are only four machines in a ring shown in Figure 4.4.

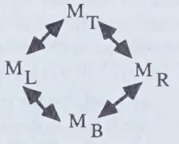


Figure 4.4 Ring of Four Machines

The arcs represent communications between the various machines. The state of each of the four machines is represented with two Boolean variables:

Machine	M_T	M_R	M_B	M_L
Position	Top	Right	Bottom	Left
Boolean State Variables	x_T, x_{Tu}	x_R, x_{Ru}	x_B, x_{Bu}	x_L, x_{Lu}

Although, in general the Boolean state variables can take on values of 1 (True) or 0 (False), Dijkstra restricts the values of the top and bottom machine's states. For the bottom machine $x_{Bu}=1$, and for the top machine $x_{Tu}=0$ by definition. These machines are therefore only two-state machines. If x is a Boolean variable let x:=0 (x:=1) denote the assignment of false (true) to x. Also, let $x:=\neg x$ denote the switching of the truth value of x. To complete the definition of the distributed system the privileges are defined with a set of rules as follows:

(i) **If** $x_B=x_R$ **and** $x_{Ru}=0$ **Then** $x_B:=\neg x_B$
(ii) **If** $x_T\neq x_L$ **Then** $x_T:=\neg x_T$

(iii) If $x_R \neq x_B$ Then $x_R := \neg x_R$ and $x_{Ru} := 1$
(iv) If $x_R = x_T$ and $x_{Ru} = 1$ Then $x_{Ru} := 0$
(v) If $x_L \neq x_T$ Then $x_L := \neg x_L$ and $x_{Lu} := 1$
(vi) If $x_L = x_B$ and $x_{Lu} = 1$ and $x_{Bu} = 0$ Then $x_{Lu} := 0$

Before the Petri net model is given a few observations are in order. First, notice that since $x_{Bu}=1$ by definition the sixth rule will never be used. Hence, there is no communication between the bottom machine and the machine to its left even in the general case where there are N+1 machines in the ring. The result is that the "ring" is not a ring at all but a string of communicating finite state machines. This does not hinder our ability to determine if the system is self stabilizing. The second observation is that the left hand side of all the rules (i)-(v) does not depend on x_{Lu}; consequently no privileges depend on it and it will not affect which moves are made in the system. It is therefore not necessary to model it, so M_L can also be considered a two-state machine. The Petri net model of the string of finite state machines is shown in Figure 4.5.

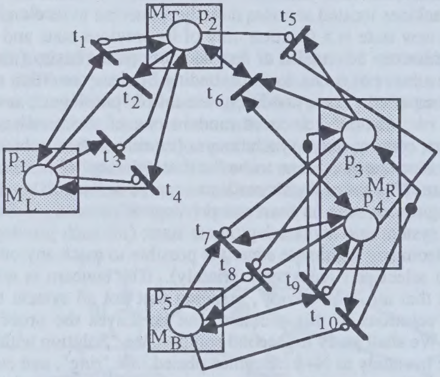


Figure 4.5 Petri Net Model of a Distributed System

The interpretation of time is that each $k \in T_s$ represents an arbitrary length time step. Notice that places were used to represent each of the machines and that $X_p(p_1,k)=x_L$, $X_p(p_2,k)=x_T$, $X_p(p_3,k)=x_{Ru}$, $X_p(p_4,k)=x_R$, and $X_p(p_5,k)=x_B$. The transitions t_j, j=1,2, ... ,10 represent the rules that describe the privileges. For instance, privilege (i) is represented with transitions t_1 and t_2. The function of the special arc —●→ was explained at the end of Section 3.0; it is used to make the assignment $x_{Ru}:=1$. The state $x_p(k)=[0\ 0\ 0\ 1\ 1]^t$ indicates that $x_B=1$ and $x_R=1$ and that the other state variables are equal to zero. As in the manufacturing system example we shall use 00011 to denote this state and other states will be denoted in a similar manner. The set of initial states $X_{po}=X_p$, the set of reachable states. Notice that there is no explicit representation of a controller. The CTL structure is formed by considering in turn each initial condition and generating the structure for that particular initial condition by firing various transitions.

The verification algorithm was used to prove that the specification $\exists p \Diamond (00111 \vee 11100)$ is true, i.e., that there exists a path from any state to the state 00111 or 11100. But notice that in these states the distributed system is deadlocked. This is easily seen by examining the above Petri net. Clearly, Dijkstra's solution is invalid since it does not satisfy the criteria necessary for the set of legitimate states. Note that the proof here is only for the case when N=3, i.e., four machines. It may be the case that the system is self stabilizing for N>3.

Acknowledgement:

The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory under contract No. 957856.

5.0 References

[1] Ben-Ari M., Pnueli A., Manna Z., "The Temporal Logic of Branching Time", Acta Inf., Vol. 20, pp. 207-226, 1983.

[2] Browne M.C., Clarke E.M., "SML - A High Level Language for the Design and Verification of Finite State Machines", IFIP WG 10.2 International Working Conference from HDL Descriptions to Guarantee Correct Circuit Designs, Grenoble, France, Sept. 1986.

[3] Clarke E.M., Emerson E.A., Sistla A.P., "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach", Tenth ACM Symp. on Principles of Programming Languages, Austin, Texas, 1983.

[4] Clarke E.M., Browne M.C., Emerson E.A., Sistla A.P., "Using Temporal Logic for Automatic Verification of Finite State Systems", Apt K.R., ed., Logics and Models of Concurrent Systems, Springer Verlag, NY, 1985.

[5] Dijkstra E.W., "Self-stabilizing Systems in Spite of Distributed Control", Comm. of the ACM, Vol. 17, No. 11, pp. 643-644, Nov. 1974.

[6] Emerson E.A., Halpern J.Y., ""Sometimes" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic", Journal of the Association for Computing Machinery, Vol. 33, No. 1, pp. 151-178, January 1986.

[7] Ichikawa A., Hiraishi K., "Analysis and Control of Discrete Event Systems Represented by Petri Nets", in Varaiya P., Kurzhanski A.B., eds., Discrete Event Systems: Models and Applications, Lecture Notes in Control and Information Sciences, Springer Verlag, NY, 1987.

[8] Inan K., Varaiya P., "Finitely Recursive Process Models for Discrete Event Systems", IEEE Trans. on Automatic Control, Vol. 33, No. 7, pp. 626-639, July 1988.

[9] Knight J.F., Passino K.M., "Decidability for Temporal Logic in Control Theory", Proc. of the Allerton Conf. on Communication, Control, and Computing, Univ. of Illinois, Urbana-Champaign, pp. 335-344, Oct. 1987.

[10] Krogh B., "Controlled Petri Nets and Maximally Permissive Feedback Logic", Proc. of the Allerton Conf. on Communication, Control, and Computing, Univ. of Illinois, Urbana-Champaign, pp. 317-326, Oct. 1987.

[11] Lamport L., ""Sometime" is Sometimes "Not Never": On the Temporal Logic of Programs", ACM Symp. on Principles of Programming Languages, pp. 174-185, 1980.

[12] Manna Z., Wolper P., "Synthesis of Communicating Processes from Temporal Logic Specifications", Dept. of Computer Science, Stanford Univ., Report No. STAN-CS-81-872, 1981.

[13] Manna Z., Pnueli A., "Verification of Concurrent Programs: a Temporal Proof System", Dept. of Computer Science, Stanford Univ., Report No. STAN-CS-83-967, 1983.

[14] Ostroff J. S., Real-Time Computer Control of Discrete Systems Modelled by Extended State Machines: A Temporal Logic Approach, PhD Dissertation, Report No. 8618, Dept. of Elect. Eng., Univ. of Toronto, Jan. 1987.

[15] Passino K.M., Antsaklis P.J., "Artificial Intelligence Planning Problems in a Petri Net Framework", Proc. of the American Control Conf., pp. 626-631, Atlanta GA, June 1988.

[16] Passino K.M., Antsaklis P.J., "Planning Via Heuristic Search in a Petri Net Framework", Proc. of the Third IEEE Int. Symposium on Intelligent Control, Arlington, VA, Aug. 1988.

[17] Peterson J.L., Petri Net Theory and the Modelling of Systems, Prentice-Hall, NJ 1981.

[18] Thistle J. G., Wonham W. M., "Control Problems in a Temporal Logic Framework", International Journal Control, Vol. 44, No. 4, pp. 943-976, 1986.

A DECOMPOSITION APPROACH FOR SENSITIVITY ANALYSIS OF DISCRETE-EVENT SYSTEMS[*]

STEPHEN G. STRICKLAND
Division of Applied Sciences, Harvard University, Cambridge, MA 02138
CHRISTOS G. CASSANDRAS
Dept. of Electrical & Computer Engineering, University of Massachusetts, Amherst, MA 01003

1 INTRODUCTION

We consider the problem of performance sensitivity/gradient estimation for general discrete-event systems (DES). Such information can be employed in classical stochastic optimization algorithms for systems where direct analytical optimization is impossible (typically the case for DES). We adopt a sample path analysis approach, that is, we assume that the system of interest is stochastic and we are given a realization of a "nominal" system (i.e. characterized by nominal parameter values). We then consider perturbations in one or more of the parameters, implicitly defining one or more perturbed systems. *Our goal is to construct, in some manner, realizations of these perturbed systems using information contained in the nominal realization.* Under mild ergodicity assumptions, we can then use these realizations to compute strongly consistent estimates of equilibrium performance sensitivities as well as unbiased, consistent estimates of transient performance sensitivities.

Note that if the structure and parameter values (nominal and perturbed) of the system are known, then we can always use discrete-event simulation to obtain the performance sensitivities of interest, albeit at often considerable computational cost. This is in fact the prevailing approach. In developing a new approach, we are interested both in reducing the computational cost and in dealing with situations where some (or all) of the stochastic parameters may be unknown (or may vary).

In simulation, construction of a sample path involves certain computations, including random number generation. Our approach is based on the idea that the "mechanism" which is generating the nominal path is performing many of the same computations; thus we can use the results embedded in the nominal path rather than repeat these computations ourselves. Our ability to do this efficiently depends to a considerable extent on the representation chosen for the DES. We use an event-based GSMP-type representation to *decompose* a sample path into a sequence of conditionally independent epochs. The problem is then to use nominal epochs to construct perturbed epochs. In some cases, we can use nominal epochs directly as perturbed epochs. In other cases, we must do a bit more work. In the following sections, we describe our DES model and outline a class of methods for constructing perturbed paths.

2 THE DES MODEL

We represent a DES by $\{S, s_0, E, D, F\}$, where S is a set of states, s_0 is a given initial state, E is a set of *event-types*, D is a *deterministic* transition function $D : S \times E \to S \cup \Lambda$[‡], and $F = \{F^e\}_{e \in E}$ is a set of probability distribution functions, associated one-to-one with the event-types, which characterizes the stochastics of the event processes in a manner described as follows.

First, let $E(s) := \{e : D(s,e) \neq \Lambda\}$ denote the *feasible set* of events at state s (also called the *event list*), that is, the set of events which can occur when the system is in state s. We view the overall event process as a superposition of renewal processes (one for each event-type) which are activated and deactivated as a function of $E(s)$, with the understanding that each activation initiates a renewal. Each F^e then corresponds to the renewal distribution of the corresponding event process.

With each *feasible* event-type, e, we associate an *age*, ξ^e, which is the elapsed time since the most recent renewal of e. Given F^e, the age defines the conditional distribution, $F^e_{x|\xi^e}$, of *residual life* associated with the event-type e. Now, given the initial state s_0, note that a realization can be represented by a sequence $\{(\tau_i, e_i)\}_{i=1,2,\ldots}$ of (interevent-time, event) pairs since we can generate the associated state sequence recursively via $s_{i+1} = D(s_i, e_i)$. Each (τ, e) pair in this sequence is a random variable whose distribution is determined by the residual life distributions of events which are feasible during the epoch. Specifically, we define the sample-path evolution of the process by the following set of dynamic equations (note that the initial state, s_0, is specified in the model description):

---

[*] This work is supported in part by the U.S. Army under contract DAAL 03-86-K-0171 and by the National Science Foundation under grant ECS-85-15449

[‡] We use Λ to denote the null value, indicating that the transition is undefined (the event cannot occur).

Initialization:

$$\begin{aligned} \xi_0^e &= 0 \text{ for all } e \in E \\ \tau_0 &= \min_{e\in E(s_0)} (x_0^e), \quad \text{where } x_0^e \sim F^e_{x|\xi_0^e} \\ e_0 &= \arg \min_{e\in E(s_0)} (x_0^e) \end{aligned} \tag{1}$$

Evolution:

$$\begin{aligned} s_{i+1} &= D(s_i, e_i) \\ \xi_{i+1}^e &= \begin{cases} \xi_i^e + \tau_i & \text{for all } e \in E(s_i) \cap E(s_{i+1}), e \neq e_i \\ 0 & \textit{otherwise} \end{cases} \\ \tau_{i+1} &= \min_{e\in E(s_{i+1})} (x_i^e), \quad \text{where } x_i^e \sim F^e_{x|\xi_i^e} \\ e_{i+1} &= \arg \min_{e\in E(s_{i+1})} (x_i^e) \end{aligned} \tag{2}$$

This model, previously used in [3], is essentially equivalent to the "age-based" GSMP formulation of [1], except that our transition function is deterministic, rather than discrete. The ages $\{\xi_e\}$ correspond to the GSMP clocks. The GSMP itself can be be viewed as an event-based formulation of the technique of supplementary variables, which converts a non-Markov process on S into a Markov process on $S \times \Re^{|E|}$. The ages (or clocks) correspond to a common choice of the supplementary variables required. Thus the process $(s(t), \{\xi_e(t)\})$, defined in the natural way, is Markov.

Note that the distribution of each (τ, e) is uniquely characterized by $\Psi := (E(s), \{\xi^e\})$, and when conditioned on Ψ, all pairs are *independent*. In the next section, we outline techniques for using this model to construct perturbed paths.

3 CONSTRUCTING PERTURBED PATHS

From (2), above, we see that construction of a perturbed path can be viewed as repeatedly iterating the procedure: (1) evaluate Ψ, and (2) generate (τ, e) using the corresponding residual life distributions. We can view this as simply the generation of random numbers from a "distribution" parameterized by Ψ. Clearly, then, the nominal path represents a source of conditionally independent random numbers which we can use to construct the perturbed path. At each iteration in the construction, we can attempt to "match" the perturbed Ψ with the value of Ψ at some point in the nominal path and, if sucessful, use the ensuing nominal (τ, e) in our perturbed construction. This "distribution matching" approach will maintain the necessary conditional independence of epochs in the perturbed path so long as each nominal (τ, e) pair is used at most once.

Practically speaking, we never have a complete nominal realization at our disposal, but only some portion of one. In particular, assume we with to construct a perturbed path concurrent, in some sense, with the evolution of the nominal path. Thus, at each nominal epoch, we compare the nominal and perturbed Ψ and, if they are equal, concatenate the nominal (τ, e) pair onto the perturbed sequence (updating Ψ). If the nominal and perturbed Ψ are not equal, we have several choices. First, we may simply wait until the next nominal epoch and try again to "match Ψs." If memory is available, we may save the current nominal (τ, e) (along with its Ψ) for possible use later in the event of other "mismatches." Second, we can explicity generate a new (τ, e) pair using the perturbed Ψ. Third, we can transform a nominal pair to alter its distribution to that corresponding to the perturbed Ψ.

For general systems, the probability of Ψ matching will be low due the the difficulty of matching the continuously distributed age component of Ψ. Thus the perturbed path may either evolve slowly (or not at all) relative to the nominal path or we may incur substantial computational expense through frequent use of options two and three, above. Cases where most (or all) of the event processes are Poisson are therefore of considerable interest, since the (τ, e) distributions are independent of the associated event ages. Thus, we need only match the $E(s)$ component of Ψ, which is generally much easier. If we permit matching at points other than event occurrence instants, then we can generally accomodate one non-Poisson process. These issues are considered in more detail in [2,3] and their references.

References

[1] Glynn, P., "A GSMP Formalism For Discrete-Event Systems," to appear *Proc. of the IEEE*, Jan. 1989.

[2] Cassandras, C.G., and Strickland, S.G., "Sample Path Properties of Timed Discrete-Event Systems," to appear in *Proc. of the IEEE*, Jan. 1989.

[3] Strickland, S.G., "Stochastic Sample-Path Analysis of Discrete-Event Dynamic Systems," Ph.D. Dissertation, University of Massachusetts, May 1988.

NEURAL COMPUTING FOR INFORMATION EXTRACTION IN CONTROL SYSTEMS

K.M. PASSINO, M.A. SARTORI, and P.J. ANTSAKLIS
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

Neural computing offers massively parallel computational facilities for the classification of patterns. In this paper we use a certain type of neural network called the "multi-layer perceptron" to classify numeric data and appropriately assign symbols to the various classes. It is shown that there are several control applications where a numeric to symbolic conversion process can be performed by the perceptron. The conversion process results in a type of "information extraction" which is similar to what is called "data reduction" in pattern recognition. After introducing the idea of using the neural network as a numeric to symbolic converter, several applications are studied. First, the relevance of the results to autonomous control is discussed. The perceptron is used as a numeric to symbolic converter for a discrete event system controller supervising a continuous variable dynamic system. Also, it is shown how the perceptron can be used to implement a fault tree which provides useful information for failure diagnosis and control purposes.

1.0 Introduction

Neural computing involves the use of neural networks to perform computations in a massively parallel fashion. A neural network is a model of the interconnected network of neurons in, for instance, a human being. Researchers hope to emulate the highly efficient "computing" that humans perform with ease. Here, we use a certain neural network called the "multi-layer perceptron" to classify numeric data and appropriately assign symbols to the various classes. This classification is a form of information extraction from the given data. The classification is performed with massively parallel computing; hence, it is inherently faster than conventional sequential processing. Several applications to control systems are discussed including discrete event systems, autonomous control systems, and fault trees. Specific examples include the use of the perceptron to convert sensor data from a surge tank (the liquid level) to symbols that are used by a discrete event controller. Also, it is shown how the perceptron can be used to implement a fault tree for an aircraft.

It must be made clear that the goal of this paper is to introduce the idea of using the multi-layer perceptron as a numeric to symbolic converter to provide measurement information for specific control purposes. There is no contribution to the theory of neural networks or pattern recognition. The work involves the application of a particular neural network to problems often encountered in some of the more recent developments in control theory. In short, the results of this paper show one alternative for implementing the interface between numeric and symbolic information, a problem of significant importance.

Generally speaking, information extraction is the transformation of detailed information into abstract, generalized information; there is always a certain loss of information in the extraction process. Here, the detailed information is an n-dimensional vector of real numbers, the numeric data. The abstract information is a set of symbols representing regions in the n-dimensional space of real numbers. The information extraction process is performed here using the multi-layer perceptron. The type of information obtained from the extraction process studied here is of the form "signal 1 is greater than 0 and less than 5, and signal 2 is greater that -3 and less than 10". The information lost in our extraction process is the exact value of the real numbers. It is understood that there are other types of information extraction. For example, the type of information extraction studied here is limited in that it does not involve characterizing the *behavior* of the system generating the data. For instance, the results will not directly perform information extraction of the form "signal 1 is the derivative of signal 2". Consequently, the results obtained in this paper are not applicable to general information extraction problems. They only apply to a special class of information extraction problems that have been encountered in recent studies in control theory that are outlined below.

The process of information extraction via the neural network studied here is closely related to what has been called "pattern recognition". In fact, the multi-layer perceptron solves what are called "deterministic classification problems" [6]. The pattern recognition process is viewed as a three step "data reduction" process [6,5]:

(i) Sensors make *measurements* of certain variables in the environment,
(ii) *Feature extraction* is then used to obtain a more global, high level description of the measured data,
(iii) *Classification algorithms* are used to name the pattern.

The viewpoint can be taken that this complete process is performed by the multi-layer perceptron. The patterns sensed are points in the n-dimensional space of real numbers. Feature extraction entails determining which region each element of the n-dimensional vector lies in. Classification involves taking logical combinations of these regions and attaching labels to them.

Rather than viewing the numeric to symbolic conversion process as a traditional pattern recognition problem, we prefer to view the multi-layer perceptron as the device which performs only step (i) (in the above process) of a more abstract recognition process in which the goal is to take some action based on the pattern detected. For step (i) of the more abstract process, the perceptron senses data in the environment and provides symbols as its measurements. These symbols are then used by a controller to make decisions on what to input into the plant. Clearly, the controller may need abstract, "symbolic" feature extraction and classification to complete the high level pattern recognition so it can make its decisions, but these problems are not studied here. A similar view of pattern recognition is taken in [23]. Note that the method used here does not employ the more complex pattern recognition techniques (e.g., statistical or syntactical) to recognize wider classes of patterns. For example, as indicated above it will not characterize the dynamical behavior of the environment over a period of time. For an overview of pattern recognition techniques see, for instance, [19].

Information extraction can be used in several different ways in control systems. The three applications studied here are described in more detail in Section 2. The first applications considered are certain discrete event systems. A sampling of references for this topic include [9,12,14,16,17,22,24]. Many others exist. Wonham and his student's work is particularly notable. In this paper we note that there are times when discrete event controllers can be used for the control of continuous variable systems. In these cases there is a need for a numeric to symbolic converter to provide the symbols to the controller. Next, for the case of autonomous control, there is a "symbolic/numeric interface" where high level, possibly intelligent controllers are used to control continuous systems [1]. The multi-layer perceptron can be used here to convert numeric information into more abstract symbolic information for the use in, for instance, an artificially intelligent planning system. Finally, the information can be extracted by the perceptron so that it is useful in higher level control decisions. For instance, the information provided by a fault tree on what the failure status of a plant is can be used to generate appropriate control actions so that the effects of the failures can be eliminated. It is shown here that the multi-layer perceptron can be used to implement a fault tree.

The theory of the multi-layer perceptron is outlined in Section 3. In Section 4, two examples are presented. First, an example of the use of the perceptron as a numeric to symbolic converter for a surge tank is given. Next, it is shown how the perceptron can be used for the implementation of a fault tree for an aircraft. Section 5 contains the references.

2.0 Information Extraction in Control Systems

In this section we discuss several ways in which information extraction is used in control theory. In practice, information is always extracted, in the modelling of processes for example. The extraction that we discuss here is the transformation of numeric data to symbolic information that is done dynamically in an on-line control system. There are three main applications to control theory which are discussed here.

The first is discrete event systems. Sometimes a plant is described by very complicated differential/difference equations. For some of these plants current control analysis and design techniques may be either very complicated, awkward, or simply inadequate. An alternative approach to analysis and design is to begin with a different model. In [14] the authors give an example of how to use a symbolic formalism, the nondeterministic finite state automaton, to describe a plant (a surge tank) that is normally described with a nonlinear differential equation. Using this description, a finite state controller was given and certain closed loop properties were verified. In these cases, for controller implementation, a numeric to symbolic converter is needed to convert the sensor data to symbols for use in the discrete event controller. The perceptron developed here is used to implement this converter.

The discrete event controllers considered in this paper will be referred to as "symbolic controllers" since they are developed using a symbolic formalism and have as inputs and outputs, symbols rather than numeric data. The situation where we have a symbolic controller controlling a continuous variable dynamic system is depicted in Figure 2.1. The symbolic controller uses Y_i (symbols) as inputs and a reference input, say R_i (symbols), and generates the control inputs to the plant U_i (symbols). The numeric/symbolic converter transforms the measurement data y(t) into symbols Y_i. The symbolic/numeric converter transforms the symbols U_i into numeric plant inputs u(t).

The numeric/symbolic converter can be thought of as analogous to analog to digital (A/D) converter in digital systems. In the one-dimensional case, the real number line (numeric data) is partitioned into regions associated with, say, binary numbers (symbols) and the values of the input are converted into the appropriate binary numbers. Indeed, there have been neural network implementations of a 4-bit A/D converter [21,7]. The work here generalizes that application. Besides performing a more general conversion

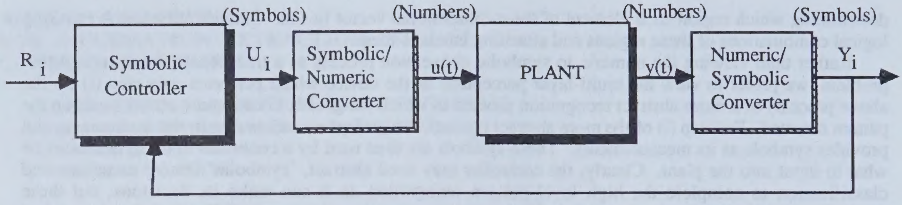


Figure 2.1 Symbolic/Numeric Conversion

process than A/D conversion, the perceptron can operate in an asynchronous mode in addition to a synchronous one. Note that the perceptron can be expected to perform numeric to symbolic conversion quickly. This is very important if it is to be placed in the control loop since delays tend to cause instability. The symbolic controller's control actions on the continuous plant can be translated to numeric inputs by a generalized method analogous to digital to analog (D/A) conversion. It may convert a symbol to a simple numeric value or to a sine wave, square wave, or some other more complex signal. This problem is not addressed here.

Autonomous controller architectures can be viewed as hierarchical with three main levels (see e.g., [1]). These are the management and organization level, the coordination level, and the execution level. The execution level contains the hardware and conventional control algorithms performing low level feedback control. The management and organization level has the interface to the operator and the intelligent learning and planning systems for supervising the actions taken at the lower levels. The coordination level provides a link between the management and organization level and the execution level. Autonomous controllers are used in the control of very complex systems that are normally hybrid. A "hybrid" system is one that contains dynamics that can be described by, for instance, differential or difference equations, and dynamics that are convenient to describe with some symbolic formalism. The controllers developed for such systems are also hybrid, with various numeric and symbolic components at each of the three levels. The numeric type algorithms are normally used at the lower levels of the architecture, the execution level, while the symbolic type algorithms are operating at the highest level, the management and organization level. There is then a symbolic/numeric interface in the autonomous controller much like the one in discrete event system theory discussed above. Consequently, the perceptron developed here will be useful in intelligent autonomous control applications.

The results here can also be used in the field of intelligent autonomous control called Expert Control. The "safety nets" in [2] use information extraction of ranges of variables for making high level control decisions in adaptive control. The results of this paper provide a method to produce the information for such control decisions. In [2] the authors propose to use an "expert controller", one that uses an expert system to emulate the actions that a human operator would perform to maintain an adaptive controller's operation. The human in the control loop is being emulated. It is then natural to use a perceptron to emulate the human's pattern classification abilities.

In general the multi-layer perceptron can be used to classify data so that control decisions can be made. The symbols attached to various regions of the n-dimensional space of real numbers may have a particular physical interpretation. The perceptron can provide failure information such as "signal 1 is in range 4 AND signal 3 is in range 5 OR signal 7 is in range 11". This information may be quite useful in making control decisions. For instance, the values of signals 1, 3, and 7 may indicate that a certain failure condition is occurring and that a specific control action ought to be taken so that the failure will not degrade system performance. Failure information of the type described above is typically generated by Fault Trees [4,3]. The *low level failures* in fault trees are characterized by regions of certain variables and fault tree *high level failures* by logical combinations of the low level failures. Here, the multi-layer perceptron determines which regions certain variables occur in and performs the appropriate ANDing and ORing to indicate what high level failure has occurred. In the following section, an outline of the theory of the multi-layer perceptron is given.

3.0 The Multi-Layer Perceptron for Information Extraction

The multi-layer perceptron is a feedforward neural network used for neural computing. The multi-layer perceptron considered here contains a *hidden layer* between the *input* and the *output layers* as illustrated in Figure 3.1. The input to the perceptron is the vector $u=[u_1\ u_2\ \dots\ u_M]^t$ (where t denotes transpose), and it contains M continuous real valued elements. The vector $x'=[x'_1\ x'_2\ \dots\ x'_{M1}]^t$, which contains M_1 binary elements, is the output of the input layer and the input to the hidden layer. The vector $x''=[x''_1\ x''_2\ \dots\ x''_{M2}]^t$, which contains M_2 binary elements, is the output of the hidden layer and the input to the output

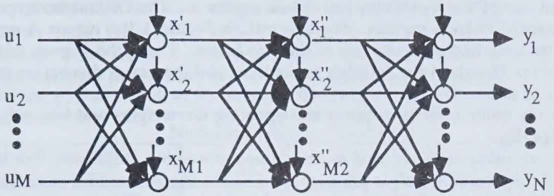
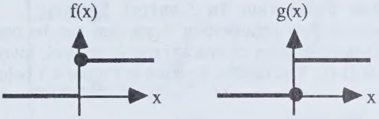


Figure 3.1 The Multi-Layer Perceptron

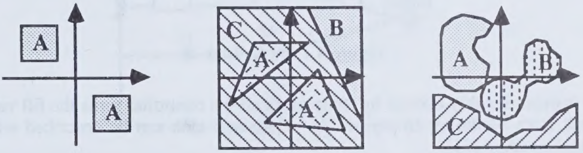
layer. The vector $y=[y_1\ y_2\ ...\ y_N]^t$, which contains N binary elements, is the output of the perceptron. In Figure 3.1, the *nodes* are denoted with circles and the *biases* with arrows that point downward. The *weights* are denoted by all of the other arrows that are between u and the input nodal layer, x' and the hidden nodal layer, and x" and the output layer. Each node produces at its output a summation of its weighted inputs and its bias which is passed through a threshold nonlinearity. The result is a binary output for each layer. Two typical threshold nonlinearities are illustrated in Figure 3.2.



(a) (b)

Figure 3.2 Threshold Nonlinearities for the Multi-Layer Perceptron

The multi-layer perceptron is commonly used for "pattern classification". In the classification problem one desires to identify certain "decision" regions that the input vector u lies in. Examples of dividing a two dimensional input space into decision regions are illustrated in Figure 3.3.



(a) (b) (c)

Figure 3.3 Examples of Two Dimensional Decision Regions

An input pattern u is presented to the perceptron. After processing, the output vector y represents certain decision regions. In particular, if u is a member of one of the decision regions, an appropriate element of the output vector y becomes a one (HI). If the input pattern u is not a member of a particular decision region, an appropriate element of the output vector y becomes a zero (LO).

The weights and biases of the multi-layer perceptron depend on the decision regions to be identified. The technique used to determine the weights and biases for the multi-layer perceptron here will be referred to as the "Harvey Method". It is based on the results presented in [8] and is outlined here. The Harvey Method can be used most effectively if the input space can be divided into decision regions composed of hypercubes, as in Figure 3.3(a), or hyperplanes, as in Figure 3.3(b). Using the Harvey Method, not only are the weights and biases determined but also the actual interconnections between the nodal layers. As with the weights and biases, the interconnections also depend on the decision regions to be identified. The three layers of the multi-perceptron, as shown in Figure 3.1, are named (from left to right) the input/plane layer, the AND layer, and the OR/output layer. The input/plane layer consists of nodes representing the hyperplanes dividing the input space. For example, in Figure 3.3(a) region A needs eight one-input input/plane nodes since the region is constrained by eight different lines. The AND layer consists of one AND node for each decision region defined by more than one hyperplane. For example, in Figure 3.3(a) region A needs two four-input AND nodes, each to denote that u is below one line, to the left of another line, above one line, and to the right of still another line. In other words, each four input AND node will indicate that u lies in one of the two boxes that define region A. The OR/output layer consists of nodes used to associate disjoint decision regions. For example, in Figure 3.3(a) region A needs one two-input OR/output node to indicate that u lies in one of the two boxes. Using the Harvey Method, the weights depend on which side of the plane the decision region lies, and the biases depend on the locations of the decision regions. As an alternative, the weights and biases could be determined by passing sample decision region data through the multi-layer perceptron and updating the weights and biases based on a training algorithm explained in [8].

There are numerous other methods used to determine the weights and biases. For instance, the Back-Propagation Training Algorithm which is presented in [18,15] can be used for perceptrons similar to the one shown in Figure 3.1, but with fully interconnected nodal layers. This algorithm may be particularly useful if the input space must to be divided into decision regions which are separated by curved borders, as in Figure 3.3(c). In [10] three other training procedures are introduced and compared to the Back-Propagation Training Algorithm. In [11], an alternative to the Back-Propagation Training Algorithm, called the "Selective Update Back-Propagation Algorithm", is introduced and shown to work in cases where the Back-Propagation Training Algorithm will not. Through the presentation of input data, the multi-layer perceptron's output, and the desired output, these algorithms train the weights and the biases until they stabilize. Although the Back-Propagation Training Algorithm seems to work for most cases, convergence of the algorithm has not yet been proven for the fully interconnected multi-layer perceptron [15,18]. Convergence for the lightly interconnected multi-layer perceptron is considered in [20].

4.0 Examples of Information Extraction In Control Systems

In this section two examples of how information extraction can be performed via the multi-layer perceptron are given. The first example, from chemical process control, involves a liquid holding "surge tank" similar to the one studied in [14]. The tank is depicted in Figure 4.1 below.

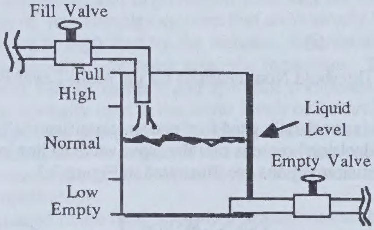


Figure 4.1 Surge Tank

The empty valve is unpredictably opened by some user and the controller turns the fill valve on and off so as to keep the tank from becoming empty or full. The surge tank can be described with the first order nonlinear differential equation [13]:

$$\frac{dh(t)}{dt}=\frac{F_i(t)}{\rho A}-\frac{kc(t)\sqrt{h(t)}}{\rho A}$$

where $\rho=99.8$, the fluid density of the liquid, $A=1$, the cross sectional area of the tank, $h(t)$ is the height of the liquid in the tank, and $k=1000$, a physical constant. The value of $c(t)$ represents the variable outflow pipe cross sectional area. The unpredictable user is modelled as $c(t)=.1c_o(t)$, where $c_o(t)$ is a random variable uniformly distributed on 0 to 1. The value of $F_i(t)= 100\sqrt{3F(t)}$, where $F(t)$ is either 1 (indicating that the fill valve is on), or 0 (indicating that the fill valve is off). Notice that with this choice if the input valve is open the liquid level will stay the same or rise.

The height of the liquid in the tank is discretized into 5 regions which are represented with the symbols y_i, $i=1,2,3,4,5$. The case when $F(t)=1$ will be represented with the symbol u_1 and $F(t)=0$ with the symbol u_0. The symbolic/numeric converter is simply a device that outputs 1 when it has as an input u_1 and 0 when its input is u_0. The controller which is the same as in [14] is given by:

Controller					
Controller Input	y_1	y_2	y_3	y_4	y_5
Controller Output	u_1	u_1	u_1	u_0	u_0

Next, the numeric/symbolic converter must be specified. Let $h_1=0$ (empty) and h_4 represent the case when the tank is completely full. Also let h_2 and h_3 represent two particular levels in the tank which distinguish, respectively, a low liquid level from a normal one and a normal level from a high one. The table below gives the necessary information to specify the classification problem.

Numeric/Symbolic Converter					
Numeric/Symbolic Converter Output	y_1	y_2	y_3	y_4	y_5
Plant Output Regions	$h(t)=h_1$	$h_1<h(t)<h_2$	$h_2\leq h(t)\leq h_3$	$h_3<h(t)<h_4$	$h(t)=h_4$

The multi-layer perceptron designed using Harvey's Method is illustrated in Figure 4.2.

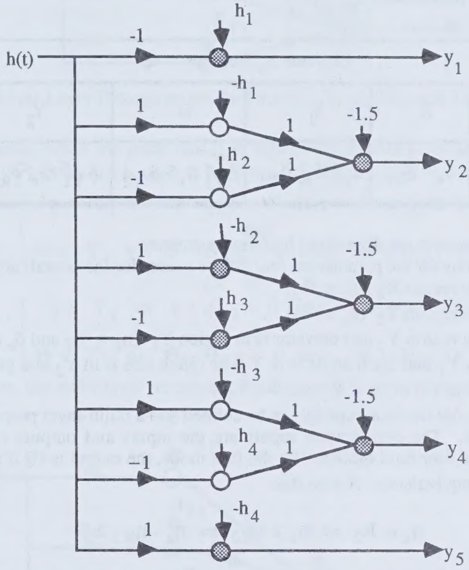


Figure 4.2 The Multi-Layer Perceptron for the Surge Tank

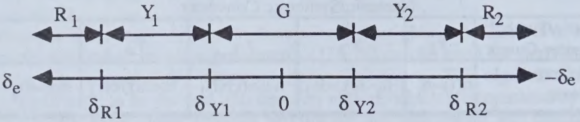
The arrows indicate amplifiers. The shaded and unshaded circles indicate a summing of the inputs to the circle (arrows pointing towards the circle) and that the sum is passed through one of the two nonlinearities shown in Figure 3.2. The shaded circles refer to Figure 3.2(a) and the unshaded circles refer to Figure 3.2(b).

The object of the controller studied in [14] was to stabilize the height of the liquid in the surge tank around a particular level. In particular, it was desired that the steady state behavior of the plant entail only a sequence of y_3 and y_4 alternating. When the plant, the numeric/symbolic converter, the controller, and the symbolic/numeric converter are connected together as in Figure 2.1 and simulated, the output of the plant behaved as desired. The plant was given two initial conditions, completely empty, and at level y_3. Each time, the controller opened the fill valve and then closed the fill valve when the level rose to y_4. The input valve was opened again when the lower level fell to y_3 and closed again when the level rose to y_4. The system continued in this fashion with the controller attempting to keep the level just above y_3. Consequently, the multi-layer perceptron successfully implemented the numeric/symbolic converter for the surge tank and its symbolic controller.

The second example involves the implementation of a fault tree for an aircraft. The multi-layer perceptron is used to indicate the failure modes which depend on the aircraft's inputs and outputs. The aircraft's input vector u and the output vector y are:

$$u = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \begin{matrix} \text{elevator (deg)} \\ \text{thrust (deg)} \end{matrix} \qquad y = \begin{bmatrix} q \\ \theta \\ \eta_z \end{bmatrix} \begin{matrix} \text{pitch rate (deg/sec)} \\ \text{pitch angle (deg)} \\ \text{load factor (g)} \end{matrix}$$

Four failure modes are considered here. To do so, each input and output is discretized into five regions with four boundaries associated with the real number line. For example, the elevator δ_e is discretized as follows:



where the G (for Green) region denotes an area of safe operation, the Y_1 and Y_2 (for Yellow) regions denote areas of warning, and the R_1 and R_2 (for Red) regions denote areas of unsafe operation. The table below defines the five regions for the elevator δ_e.

Elevator δ_e Regions					
Elevator Region	R_1	Y_1	G	Y_2	R_2
Elevator Range	$\delta_e \le \delta_{R1}$	$\delta_{R1} < \delta_e \le \delta_{Y1}$	$\delta_{Y1} < \delta_e < \delta_{Y2}$	$\delta_{Y2} \le \delta_e < \delta_{R2}$	$\delta_{R2} \le \delta_e$

The four other aircraft parameters are discretized in a similar manner.

Using the defined regions for the parameters, four failure modes for the aircraft are identified as follows:

1) load factor is in region R_2 ($\eta_z \in R_2$)
2) load factor is in region Y_2 ($\eta_z \in Y_2$)
3) load factor is in region Y_2 and elevator is in region Y_1 ($\eta_z \in Y_2$ and $\delta_e \in Y_1$)
4) (pitch rate is in Y_1 and pitch angle is in Y_1) or (pitch rate is in Y_2 and pitch angle is in Y_2) [($q \in Y_1$ and $\theta \in Y_1$) or ($q \in Y_2$ and $\theta \in Y_2$)]

For the four failure modes decision regions can be defined and a multi-layer perceptron can be designed using the Harvey Method. The perceptron's inputs are the inputs and outputs of the aircraft and the perceptron's outputs are the four fault modes. For the first mode, the output is HI if the load factor η_z is in region R_2. Let $\Leftrightarrow$ denote equivalence. Notice that

$$\eta_z \in R_2 \Leftrightarrow \eta_z \ge \eta_{R2} \Leftrightarrow \eta_z - \eta_{R2} \ge 0$$

Using the last equation, the multi-layer perceptron for the first fault mode is shown at the top of Figure 4.3. For the second fault mode, the output should be HI if the load factor η_z is in region Y_2. Notice that

$$\eta_z \in Y_2 \Leftrightarrow \eta_{Y2} \le \eta_z < \eta_{R2} \Leftrightarrow \begin{cases} \eta_z \ge \eta_{Y2} \Leftrightarrow \eta_z - \eta_{Y2} \ge 0 \\ \text{and} \\ \eta_z < \eta_{R2} \Leftrightarrow -\eta_z + \eta_{R2} > 0 \end{cases}$$

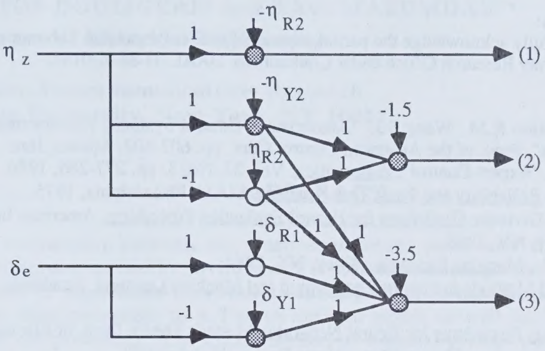
Using the last equations, the multi-layer perceptron for the second fault mode is shown in the middle of Figure 4.3. For the third fault mode, the output should be HI if the load factor η_z is in region Y_2 and the elevator δ_e is in region Y_1. Notice that

$$\eta_z \in Y_2 \Leftrightarrow \eta_{Y2} \le \eta_z < \eta_{R2} \Leftrightarrow \begin{cases} \eta_z \ge \eta_{Y2} \Leftrightarrow \eta_z - \eta_{Y2} \ge 0 \\ \text{and} \\ \eta_z < \eta_{R2} \Leftrightarrow -\eta_z + \eta_{R2} > 0 \end{cases}$$

and

$$\delta_e \in Y_1 \Leftrightarrow \delta_{R1} < \delta_e \le \delta_{Y1} \Leftrightarrow \begin{cases} \delta_e > \delta_{R1} \Leftrightarrow \delta_e - \delta_{R1} > 0 \\ \text{and} \\ \delta_e \le \delta_{Y1} \Leftrightarrow -\delta_e + \delta_{Y1} \ge 0 \end{cases}$$

Using the last equations, the multi-layer perceptron for the third fault mode is shown in Figure 4.3.



η_z
δ_e
$-\eta_{R2}$
$-\eta_{Y2}$
η_{R2}
$-\delta_{R1}$
δ_{Y1}
1
-1
-1.5
-3.5
(1)
(2)
(3)

Figure 4.3 The Multi-Layer Perceptron for the First (1), Second (2), and Third (3) Fault Modes

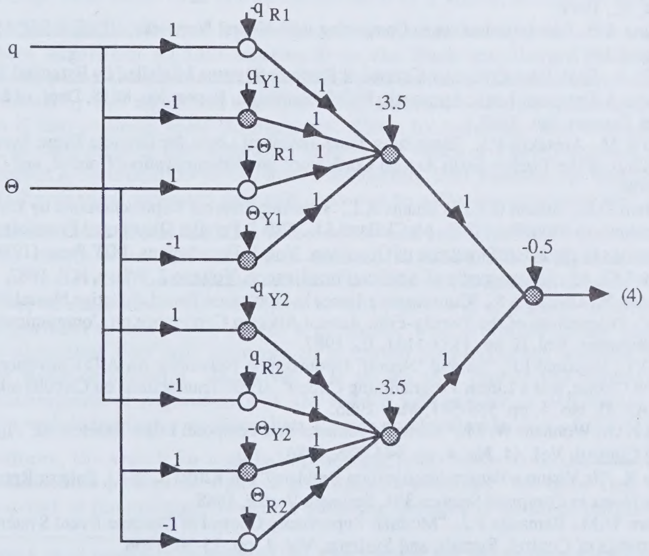
For the fourth fault mode, either the pitch rate is in region Y_1 and the pitch angle is in region Y_1 or the pitch rate is in region Y_2 and the pitch angle is in region Y_2. Notice that

$$\left\{\begin{array}{c} q \in Y_1 \Leftrightarrow q - q_{R1} > 0 \text{ and } -q + q_{Y1} \geq 0 \\ \text{and} \\ \Theta \in Y_1 \Leftrightarrow \Theta - \Theta_{R1} > 0 \text{ and } -\Theta + \Theta_{Y1} \geq 0 \end{array}\right.$$

$$\text{or}$$

$$\left\{\begin{array}{c} q \in Y_2 \Leftrightarrow q - q_{R2} > 0 \text{ and } -q + q_{Y2} \geq 0 \\ \text{and} \\ \Theta \in Y_2 \Leftrightarrow \Theta - \Theta_{R2} > 0 \text{ and } -\Theta + \Theta_{Y2} \geq 0 \end{array}\right.$$

Using the last equations, the multi-layer perceptron for the fourth fault mode is shown in Figure 4.4.



q
Θ
$-q_{R1}$
q_{Y1}
$-\Theta_{R1}$
Θ_{Y1}
$-q_{Y2}$
q_{R2}
$-\Theta_{Y2}$
Θ_{R2}
1
-1
-3.5
-0.5
(4)

Figure 4.4 The Multi-Layer Perceptron for the Fourth (4) Fault Mode

The aircraft and the multi-layer perceptron which implemented the fault tree were simulated. The multi-layer perceptron appropriately identified all of the fault modes.

Acknowledgements:

The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory under Contract No. 957856 and the Army Research Office under Contract No. DAAL 03-88-K-0144.

5.0 References

[1] Antsaklis P.J., Passino K.M., Wang S.J., "Autonomous Control Systems: Architecture and Fundamental Issues", Proc. of the American Control Conf. pp. 602-607, Atlanta, June 1988.

[2] Astrom K.J., et al, "Expert Control", Automatica, Vol. 22, No. 3, pp. 277-286, 1986.

[3] Barlow R.E., et al, Reliability and Fault Tree Analysis, SIAM, Philadelphia, 1975.

[4] Battelle Columbus Division, Guidelines for Hazard Evaluation Procedures, American Institute of Chemical Engineers, NY, 1985.

[5] Forsyth R., Rada R., Machine Learning, Wiley, NY, 1986.

[6] Fu K.S., Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, NY, 1968.

[7] Gray D.L., Synthesis Procedures for Neural Networks, Master's Thesis, Dept. of Electrical and Computer Eng., University of Notre Dame, Notre Dame, IN, July 1988.

[8] Harvey R.L., "Multi-layer Perceptron Theory", Dept. of Electrical and Computer Engineering Seminar, University of Notre Dame, March 21, 1988

[9] Ho Y.C., Li S., "Extensions of Infinitesimal Perturbation Analysis", Trans. on Automatic Control, Vol. 33, No. 5, pp. 427-438, May 1988.

[10] Huang W.Y., Lippmann R.P., "Neural Net and Traditional Classifiers", Presented at the IEEE Conference on Neural Information Processing Systems - Natural and Synthetic, Denver, CO, November 1987.

[11] Huang S.C., Supervised Learning with a Selective Update Strategy for Artificial Neural Networks, Master's Thesis, Dept. of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, IN, Aug. 1988.

[12] Inan K., Varaiya P., "Finitely Recursive Process Models for Discrete Event Systems", IEEE Transactions on Automatic Control, Vol. 33, No. 7, pp. 626-639, July 1988.

[13] Johnson A., Process Dynamics Estimation and Control, Peregrinus Pub., London, 1985.

[14] Knight J.F., Passino K.M., "Decidability for Temporal Logic in Control Theory", Proceedings of the Twenty-Fifth Annual Allerton Conference on Communication, Control, and Computing, Vol. I, pp. 335-344, IL, 1987.

[15] Lippmann R.P., "An Introduction to Computing with Neural Networks", IEEE ASSP Magazine, pp. 4-22, April 1987.

[16] Ostroff J. S., Real-Time Computer Control of Discrete Systems Modelled by Extended State Machines: A Temporal Logic Approach, PhD Dissertation, Report No. 8618, Dept. of Elect. Eng., Univ. of Toronto, Jan. 1987.

[17] Passino K.M., Antsaklis P.J., "Branching Time Temporal Logic for Discrete Event System Analysis", Proceedings of the Twenty-Sixth Annual Conference on Communication, Control, and Computing, Sept. 1988.

[18] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning Internal Representations by Error Propagation", in Rumelhart D.E., McClelland J.L. (Eds.), Parallel Distributed Processing: Explorations in the MicroStructrure of Cognition, Vol. 1: Foundations, MIT Press (1986).

[19] Shapiro S.C., ed., Encyclopedia of Artificial Intelligence, Volume 2, Wiley, NY, 1987.

[20] Shrivatava Y., Dasgupta S., "Convergence Issues in Perceptron Based Adaptive Neural Network Models", Proceedings of the Twenty-Fifth Annual Allerton Conference on Communication, Control, and Computing, Vol. II, pp. 1133-1141, IL, 1987.

[21] Tank D.T., Hopfield J.J., "Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Transactions on Circuits and Systems, Vol. CAS-33, No. 5, pp. 533-541, May 1986.

[22] Thistle J. G., Wonham W. M., "Control Problems in a Temporal Logic Framework", International Journal Control, Vol. 44, No. 4, pp. 943-976, 1986.

[23] Wilson R., "Is Vision a Pattern Recognition Problem?", in Kittler J. (Ed.), Pattern Recognition, Lecture Notes in Computer Science 301, Springer-Verlag, 1988.

[24] Wonham W.M., Ramadge P.J., "Modular Supervisory Control of Discrete-Event Systems", Mathematics of Control, Signals, and Systems, Vol. 1, pp. 13-30, 1988.

MORE ON PARETO OPTIMAL FLOW CONTROL*

CHRISTOS DOULIGERIS and RAVI MAZUMDAR**
Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University, New York, NY 10027

ABSTRACT

The connection between the Nash equilibrium point and the Pareto optimal set in a computer networks environment is discussed through the existence and the convergence of algorithms that lead to these points. A modification to a proposed algorithm that converges to a Pareto optimal point as well as limitations of this algorithm are presented. The maximization of a global system power defined as the product of individual powers is shown to give a Pareto optimal point and steers us to the introduction of a decentralized algorithm for the computation of the optimal point. Properties of this algorithm are also discussed.

I. INTRODUCTION

In the exercise of flow control in a computer networks environment a decentralizable algorithm to achieve a performance objective is desirable. By decentralizability it is meant that decisions can be made locally at a certain node or by an individual user by using only locally available information.

In [13], Jaffe pointed out that there are performance criteria that by their nature are non decentralizable, i.e. there exists no decentralizable algorithm that achieves the optimum solution. In particular it was shown that power - defined as the weighted ratio of the throughput over the average time delay -was not decentralizable in a virtual circuit network environment.

In [6], Cansever, motivated by the above mentioned work, presented an algorithm (to be called here Algorithm A) that starting from the Nash equilibrium point converged to a Pareto optimal point, if some conditions on the performance criterion were satisfied. The key property utilized by this algorithm was that all users could be better off compared to the Nash equilibrium if they reduced their throughputs. Thus, by reducing their throughputs by equal amounts ϵ at each step of the algorithm they could reach a Pareto optimal point. In this work we present a modification to the above algorithm, that enables the users to reduce their throughputs by unequal amounts and still converge to a Pareto optimal point.

In [9], the authors presented a set of necessary and sufficient conditions that the throughputs have to satisfy when the network is reduced to a link shared by a number of virtual circuits, so as to have a Pareto optimal point. The cost criterion used for this characterization was the power of each class. The structure of this result will enable us to identify the points to which Algorithm A converges, to get a geometric interpretation of the algorithm and to propose an algorithm that will converge to a Pareto optimal point without having to pass through the Nash equilibrium. Moreover it is shown that one of the assumptions made to prove the convergence of Algorithm A is not valid if we consider as performance criterion the power with a weighting factor greater than 1, even in the simple case of the M/M/1 queue.

Furthermore, the search for a global optimizing point leads us to the consideration of the product of individual powers as a global performance measure. We show that any extremum or inflexion point of the system power in a queueing network is a Pareto optimal point. In the

---

* This work was partially supported by the Center for Telecommunications Research at Columbia University under the National Science Foundation Grant #CDR-84-21402.

** On leave at INRS-Télécommunications, Montreal, Canada.

special case of a M/M/1 queue it is shown that this is the unique maximization point and can be reached via a decentralized algorithm. The algorithm is of the type presented in [3][8][11] but now each user maximizes the system power and not its own power. We have a cooperative solution. Convergence properties of the algorithm are discussed in relation with the previous work in [8][10][4][5].

The paper is organized as follows. In section II the general model is presented and the Nash and Pareto optimal points are formally defined. In section III the algorithm proposed by Cansever is presented as well as a modification to achieve a set of Pareto optimal points. In section IV, using the power as the cost criterion, it is shown that a generalized power does not satisfy the concavity property necessary for the convergence of Algorithms A and B. Using the structure of the optimal points two other algorithms are introduced. In section V a global performance measure is defined as the product of the individual powers. It is shown that maximization of the system power leads to a Pareto optimal point. A decentralized algorithm is proposed and its properties are studied. Section VI concludes the paper.

II. THE MODEL AND THE OPTIMAL SOLUTIONS

Consider a computer communication network, let it be a packet switched or a circuit switched one, modeled as a queueing network, where n users share its facilities and compete for the available resources. Let γ_i be the throughput of user i and D_i its average time delay. Let Γ_i be the set of feasible throughputs for user i, i.e. the set of throughputs that does not saturate any queue of the network and $\Gamma = (\Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_n)$. Each user consists a separate class of customers and has its own performance objective. We assume that a user tries to maximize $h_i(\gamma_1, \ldots, \gamma_n)$, where function h_i has some desirable properties that will be mentioned later. If each user tries to maximize its own performance without cooperation with the other users we get an equilibrium point where each user would be worse off if it unilaterally changed its throughput, i.e. we reach a Nash equilibrium. If we want a globally efficient solution, i.e. a point that if any user deviated at least one user would be worse off, then a Pareto optimal point is needed. We present a formal definition of the Nash and the Pareto optimal points [1].

Definition 1. Let $h_i(\gamma_1, \gamma_2, \ldots, \gamma_{i-1}, \gamma_i, \gamma_{i+1}, \ldots, \gamma_n)$, be the performance objective of user i, $i = 1, 2, \ldots, n$. Then an admissible point $(\gamma_1^*, \gamma_2^*, \ldots, \gamma_n^*)$ is a Nash equilibrium solution iff

$$h_1(\gamma_1^*, \gamma_2^*, \ldots, \gamma_n^*) > h_1(\gamma_1, \gamma_2^*, \ldots, \gamma_n^*) \qquad \forall \gamma_1 \in \Gamma_1,$$

$$\vdots$$

$$h_i(\gamma_1^*, \ldots, \gamma_i^*, \ldots, \gamma_n^*) > h_i(\gamma_1^*, \ldots, \gamma_i, \ldots, \gamma_n^*) \quad \forall \gamma_i \in \Gamma_i,$$

$$\vdots$$

$$h_n(\gamma_1^*, \gamma_2^*, \ldots, \gamma_n^*) > h_n(\gamma_1^*, \gamma_2^*, \ldots, \gamma_n) \qquad \forall \gamma_n \in \Gamma_n,$$

where $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$ are the sets of all feasible throughputs for users 1,2, ..., n, respectively.

Definition 2. A point $\tilde{\gamma} \in \Gamma$ is called Pareto optimal iff for any $(\tilde{\gamma} + \delta\gamma) \in \Gamma$ either

$$h_i(\tilde{\gamma} + \delta\gamma) - h_i(\tilde{\gamma}) = 0 \quad \forall \quad 1 \leq i \leq n$$

or for at least one i

$$h_i(\tilde{\gamma} + \delta\gamma) - h_i(\tilde{\gamma}) < 0.$$

III. PARETO OPTIMAL ALGORITHMS

In [6], Cansever, exploiting several properties of the cost function, proved that under relatively general conditions all users can be better off compared to the Nash equilibrium if they reduce their throughputs. An algorithm was presented that led to a Pareto optimal point. In the sequel we will call this algorithm, algorithm A.

The main assumptions underlying this algorithm are:

1. $h_i(.,.)$ is twice continuously differentiable in its arguments,
2. $\partial h_i(., D_i(\gamma))/\partial \gamma_i > 0$; and $\partial h_i(\gamma_i, .)/\partial D_i < 0$.
3. h_i is strictly concave.

We present the algorithm as well as a modification to this algorithm for the 2-users case. Generalization to the $n-$users case is straightforward.

Algorithm A

Start from the Nash equilibrium point and update as follows:

$$\gamma_i^{k+1} = \gamma_i^k - \epsilon, \text{ if } \frac{\{\partial h_i}{\partial \gamma_i} + \frac{\partial h_i}{\partial D_i}\left(\frac{\partial D_i}{\partial \gamma_i} + \frac{\partial D_i}{\partial \gamma_j}\right)\}|_{(\gamma_i^k, \gamma_j^k)} < 0;$$

$$\gamma_i^{k+1} = \gamma_1^k, \text{ if } \{\frac{\partial h_i}{\partial \gamma_i} + \frac{\partial h_i}{\partial D_i}\left(\frac{\partial D_i}{\partial \gamma_i} + \frac{\partial D_i}{\partial \gamma_j}\right)\}|_{(\gamma_i^k, \gamma_j^k)} \geq 0; \tag{1}$$

$$\gamma_j^{k+1} = \gamma_j^k - \epsilon, \text{ if } \gamma_i^{k+1} = \gamma_i^k - \epsilon;$$

$$\gamma_j^{k+1} = \gamma_j^k, \quad \text{ if } \gamma_i^{k+1} = \gamma_i^k;$$

where $\epsilon > 0$ is sufficiently small, $i, j = 1, 2$ and it has been assumed that the performance criterion of user i is the first to satisfy equation (1).

In most cases the Pareto optimal point is not unique, but there is a set of points that are Pareto optimal. Algorithm A converges to one of these points. At this point all users have an equal distance from the Nash equilibrium point and achieve a greater power than the one at the Nash equilibrium. By reducing their throughputs they achieve a better tradeoff between throughput and time delay expressed as a larger value for their performance objective. By a slight modification of the algorithm a set of Pareto optimal points can be reached. These Pareto points correspond to operating points where each user is better off compared with the Nash equilibrium but the throughputs at these points are not equally distanced from the throughputs at the Nash equilibrium.

Algorithm B

Start from the Nash equilibrium point and update as follows:

$$\gamma_i^{k+1} = \gamma_i^k - \epsilon_i, \text{ if } \{\frac{\partial h_i}{\partial \gamma_i} + \frac{\partial h_i}{\partial D_i}\left(\frac{\partial D_i}{\partial \gamma_i} + \frac{\partial D_i}{\partial \gamma_j}\right)\}|_{(\gamma_i^k, \gamma_j^k)} < 0;$$

$$\gamma_i^{k+1} = \gamma_1^k, \text{ if } \{\frac{\partial h_i}{\partial \gamma_i} + \frac{\partial h_i}{\partial D_i}\left(\frac{\partial D_i}{\partial \gamma_i} + \frac{\partial D_i}{\partial \gamma_j}\right)\}|_{(\gamma_i^k, \gamma_j^k)} \geq 0;$$

$$\gamma_j^{k+1} = \gamma_j^k - \epsilon_j, \text{ if } \gamma_i^{k+1} = \gamma_i^k - \epsilon;$$

$$\gamma_j^{k+1} = \gamma_j^k, \quad \text{ if } \gamma_i^{k+1} = \gamma_i^k;$$

where $\epsilon_i > 0$ and$\epsilon_j > 0$ are sufficiently small and not necessarily equal, $i, j = 1, 2$.

The following lemma establishes the convergence properties of Algorithm B.

Lemma 1. *If the same assumptions as in Algorithm A hold, then Algorithm B converges to a point $(\tilde{\gamma}_1, \tilde{\gamma}_2)$, which is Pareto optimal and at this point each user is better off compared to the Nash equilibrium.*

Proof : The proof follows similar lines as the proof of convergence of Algorithm A and is omitted.

●

Algorithm B apart for giving a set of Pareto optimal points rather than a single point, is also attractive because there is no need for both users to have the same step size ϵ. A graphical interpretation of both algorithms is given in section IV.

IV. THE SINGLE QUEUE CASE

In this section we consider a set of n-users sharing the resources of a single queue. The structure of the problem gives us the possibility of a complete characterization of both the Nash equilibrium and the Pareto optimal set.

The motivation behind algorithm A and its extension in algorithm B was its potential usage in a virtual circuit environment with power as the performance criterion. Power is defined as the weighted ratio of throughput over the average time delay [3][8]. For the case when we have a single link shared by n users power of user i is given by

$$P_i = \frac{\gamma_i^{\beta_i}}{D_i} = \gamma_i^{\beta_i}(\mu - \gamma_1 - \ldots - \gamma_n),$$

where $\gamma_1, \gamma_2, \ldots, \gamma_n$ are the throughputs of the users , μ the service rate and β_i the weighting factor.

If $\beta_i = 1$ for all i, it can be easily checked that power satisfies all the assumptions of Algorithm A. But for $\beta_i > 1$ concavity is no longer present as the following counterexample shows.

Example : Consider the case of 2 users sharing the resources of an exponential server of rate μ. Then $P_1 = \gamma_1^{\beta_1}(\mu - \gamma_1 - \gamma_2)$,

$$\frac{\partial P_1}{\gamma_1} = \beta_1(\mu - \gamma_2)\gamma_1^{\beta_1 - 1} - (\beta_1 + 1)\gamma_1^{\beta_1},$$

$$\frac{\partial^2 P_1}{\partial^2 \gamma_1} = \beta_1 \gamma_1^{\beta_1 - 2}[(\beta_1 - 1)(\mu - \gamma_2) - (\beta_1 + 1)\gamma_1].$$

If $\beta_1 > 1$, then for $\gamma_1 > \frac{(\beta_1 - 1)(\mu - \gamma_2)}{\beta_1 + 1}$ P_1 is concave and is convex otherwise.

A similar result can be obtained for the $n-$users case.

•

The nonconcavity of the power function for $\beta_i > 1$ does not allow the use of algorithms A and B to achieve a Pareto optimal point. Nevertheless, the structure of the Nash equilibrium and the Pareto optimal set can be completely characterized. See a characterization in [8] and [9]. We present without proof the main results from [8] and [9] and use these results to propose Pareto optimal algorithms for the case of a single queue.

Proposition 1. *With n classes of users having weighting factors $\beta_i|_{i=1}^{n}$, the Nash equilibrium point is given by*

$$\gamma_i^* = \frac{\beta_i \mu}{1 + \sum_{k=1}^{n} \beta_k} \qquad \text{for} \qquad i = 1, \ldots, n. \tag{2}$$

Proposition 2. *A necessary and sufficient condition for a set $\bar{\gamma}$ of throughputs to be Pareto optimal is*

$$\sum_{i=1}^{n}(\beta_i + 1)\bar{\gamma}_i \prod_{\substack{k=1 \\ k \neq i}}^{n} \beta_k = \prod_{k=1}^{n} \beta_k \, \mu \tag{3}$$

Given the structure of the two equilibria a geometric interpretation of algorithms A and B can be easily seen in figures 1-4 for the 2 and 3 users cases.

Proposition 3.

The Pareto optimal set divides the space of feasible solutions into two subspaces. The Nash equilibrium is never in the same subspace with point (0,0, ...,0).

Proof : Let

$$f(\gamma) = \sum_{i=1}^{n} \prod_{\substack{k=1 \\ k \neq i}}^{n} \beta_k (\beta_i + 1)\gamma_i = \prod_{k=1}^{n} \beta_k \mu. \tag{4}$$

Note that $f(\tilde{\gamma}) = 0$, i.e. this function characterizes the Pareto optimal set.

Then, $f(0, \ldots, 0) = -\prod_{k=1}^{n} \beta_k \mu < 0$

and $f(\text{Nash}) = \sum_{i=1}^{n} \prod_{\substack{k=1 \\ k\neq i}}^{n} \beta_k (\beta_i + 1) \dfrac{\beta_i \mu}{1 + \sum_{i=1}^{n} \beta_i} - \prod_{k=1}^{n} \beta_k \mu =$

$$\prod_{i=1}^{n} \beta_k \mu \left[\frac{\sum_{i=1}^{n} (\beta_i + 1) - 1 + \sum_{i=1}^{n} \beta_i}{1 + \sum_{i=1}^{n} \beta_i} \right] > 0.$$

Once we know the structure of the Pareto optimal set and the position of the Nash equilibrium with respect to that set we are able to propose an algorithm similar to Algorithms A and B, that starts from the Nash equilibrium and converges to a Pareto point without having to satisfy all the conditions that Algorithms A and B have to satisfy.

Algorithm C

Starting from the Nash equilibrium point users reduce their throughputs by small amounts ϵ_i, $i = 1, \ldots, n$ *till they reach the Pareto optimal set.*

Note that the above algorithm provides a Pareto optimal point with the property that all users are better off compared to the Nash equilibrium point. Algorithm C is almost identical to algorithm B except from the terminating condition, which now does not depend on the concavity of the cost function but on the structure of the Pareto optimal set. Points close to the Nash equilibrium could also be used as starting points as long as the starting point is on the same side as the Nash point. From the geometry of the optimal points and their relevant position in the space of feasible throughputs it can be seen that Algorithm C always converges to nonzero throughputs. From figures 3 and 4 we can see that the Pareto optimal points that algorithms B and C converge belong to the segment AB or the triangle ABC for the 2 and 3 users cases respectively.

The proposed algorithms A, B, C, have the desirable characteristic that all users are better off compared to the Nash point but this point has to be reached first in order to proceed to the execution of the algorithm. Convergence to the Nash equilibrium is not always possible in a finite number of steps and thus only algorithm C seems to be rather robust to the exact choice of the initial point.

The fact that the space is divided in two subspaces by the Pareto optimal set with the properties shown in Proposition (3) suggests a rather robust algorithm that converges to an efficient point but that does not guarantee a non zero throughput for all users.

Algorithm D

Let a feasible $(\gamma_1^0, \ldots, \gamma_n^0)$ *be any starting point.*

i) Check if the point belongs to the subspace that includes the origin (case 1) or not (case 2), by checking the sign of $f(\gamma_1^k, \ldots, \gamma_n^k)$.

ii) If case 1, then : increase all throughputs by small amounts ϵ_i.

If case 2, then : reduce all throughputs by small amounts ϵ_i.

iii) Check if the Pareto surface has been reached by checking the sign of $f(\gamma_1^k, \ldots, \gamma_n^k)$.
If yes : STOP; this is a Pareto optimal point
If no: Go back to step (ii).

The main features of this algorithm is its independence from the calculation of the Nash equilibrium. However, the algorithm proposed for the calculation of the Nash equilibrium could be used to get a starting point that would be closer to the Nash equilibrium and most probably would not lead to any zero throughputs for any of the users. Another natural choice for starting point is the point where each user maximizes its own power regardless of the presence of the other users.

Eventhough the proposed algorithms lead to a Pareto optimal point, they are closely related to the Nash equilibrium or to the structure of the optimal set. Algorithm D can converge to a zero throughput for one or more of the users an undesirable fact in most applications. The issue of global optimality and of a decentralized algorithm that leads to a Pareto optimal point is discussed in the next section.

V. A GLOBAL POWER MEASURE

Given the individual performance objectives, a global performance objective is very often defined as the sum or the weighted sum of the individual objectives[14],[2]. The appropriateness of such a definition has been the point of discussion of several authors [2][3]. Especially for flow control power various definitions of a global performance measures are presented in [3]. The sum of individual powers and the product of powers seem to have the most desirable characteristics, especially in the case of different weighting factors. The special feature of the product of powers that its maximization never leads to zero throughputs for some of the user led to its adoption by Yemini [15]. Although it is known that if a point maximizes the sum of the individual powers is a Pareto optimal point [9], no connection of the product of powers and the Pareto optimal points has been done so far. Such a connection is established by the following proposition.

Proposition 4. *Any extremum or inflexion point of the system power defined as the product of powers is a Pareto optimal point.*

Proof : Let $P_{\text{sys}} = P_1 \ldots P_n$. If $\tilde{\gamma} = (\tilde{\gamma}_1, \ldots, \tilde{\gamma}_n)$ is an extremum point of P_{sys} then we must have

$$\left.\frac{\partial P_{\text{sys}}}{\partial \gamma_i}\right|_{\tilde{\gamma}} = 0, \qquad i = 1, \ldots, n.$$

$$\left.\frac{\partial P_{\text{sys}}}{\partial \gamma_1}\right|_{\tilde{\gamma}} = 0 \Rightarrow [\frac{\partial P_1}{\partial \gamma_1} \textstyle\prod_{k=2}^n P_k + \ldots + \frac{\partial P_n}{\partial \gamma_1} \prod_{k=1}^{n-1} P_k]_{\tilde{\gamma}} = 0$$

$$\vdots$$

$$\left.\frac{\partial P_{\text{sys}}}{\partial \gamma_n}\right|_{\tilde{\gamma}} = 0 \Rightarrow [\frac{\partial P_1}{\partial \gamma_n} \textstyle\prod_{k=2}^n P_k + \ldots + \frac{\partial P_n}{\partial \gamma_n} \prod_{k=1}^{n-1} P_k]_{\tilde{\gamma}} = 0$$

Equivalently we can write, $\begin{bmatrix} \frac{\partial P_1}{\partial \gamma_1} & \cdots & \frac{\partial P_n}{\partial \gamma_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial P_1}{\partial \gamma_n} & \cdots & \frac{\partial P_n}{\partial \gamma_n} \end{bmatrix} \begin{bmatrix} \prod_{k=2}^n P_k \\ \vdots \\ \prod_{k=1}^{n-1} P_k \end{bmatrix}_{\tilde{\gamma}} = 0$. This implies that the matrix has a non empty null space, which is equivalent to:

$$\left|\begin{matrix} \frac{\partial P_1}{\partial \gamma_1} & \cdots & \frac{\partial P_n}{\partial \gamma_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial P_1}{\partial \gamma_n} & \cdots & \frac{\partial P_n}{\partial \gamma_n} \end{matrix}\right|_{\tilde{\gamma}} = 0$$

But the last equation is precisely the necessary condition for a point to be Pareto optimal [2][7]. The sufficiency is easily proven by noting that at each of the equations that define the extremum points, $P_1, \ldots, P_n$ are greater than zero which implies that there is a change in the signs of the partial derivatives, i.e. there can be no deviation from that point without rendering one of the users worse off.

•

The above proposition works for a general network. In the case of a simple M/M/1 queue an even stronger result can be derived that will allow the implementation of a decentralized algorithm to achieve a Pareto optimal solution.

Proposition 5. *The system power defined as the product of individual powers in the M/M/1 case has a unique extremum point, which is a maximization point.*

Proof :

$$P_{\text{sys}} = P_1 \ldots P_n = \prod_{i=1}^{n} \gamma_i^{\beta_i} (\mu - \gamma_1 - \ldots - \gamma_n)$$

$$\frac{\partial P_{\text{sys}}}{\partial \gamma_i}|_{\gamma = \tilde{\gamma}} = 0 \Leftrightarrow$$

$$\beta_i \tilde{\gamma}_i^{\beta_i - 1} (\mu - \tilde{\gamma}_1 - \ldots - \tilde{\gamma}_n)^n \prod_{\substack{k=1 \\ k \neq i}}^{n} \tilde{\gamma}_i^{\beta_i} - n(\mu - \tilde{\gamma}_1 - \ldots - \tilde{\gamma}_n)^{n-1} \prod_{i=1}^{n} \tilde{\gamma}_i^{\beta_i} = 0$$
$$\Leftrightarrow \beta_i (\mu - \tilde{\gamma}_1 - \ldots - \tilde{\gamma}_n) - n\tilde{\gamma}_i = 0 \tag{5}$$

For $i = 1, 2, \ldots, n$ we get linear system of equations with unique solution

$$\tilde{\gamma}_i = \frac{\beta_i \mu}{n + \sum_{k=1}^{n} \beta_k}. \tag{6}$$

To check if this point is indeed a maximal point knowledge of the second order derivatives is necessary.

Let $D_i = \frac{\partial P_{\text{sys}}}{\partial \gamma_i}$, $D_{ii} = \frac{\partial P_{\text{sys}}^2}{\partial \gamma_1^2}$ and $D_{ij} = \frac{\partial^2 P_{\text{sys}}}{\partial \gamma_i \partial \gamma_j}$.

Because $D_i|_{\tilde{\gamma}} = 0$ we get

$$D_{ii}|_{\tilde{\gamma}} = -\tilde{\gamma}_i^{\beta_i - 1} (\beta_i + n)(\mu - \tilde{\gamma}_1 - \ldots - \tilde{\gamma}_n)^{n-1} \prod_{\substack{k \neq i \\ k=1}}^{n} \tilde{\gamma}_k^{\beta_k}$$

$$D_{ij}|_{\tilde{\gamma}} = -\beta_i \tilde{\gamma}_i^{\beta_i - 1} (\mu - \gamma_1 - \ldots - \gamma_n)^{n-1} \prod_{\substack{k \neq i \\ k=1}}^{n} \gamma_k^{\beta_k}$$

Let $\Delta = \det[D_{ij}(\tilde{\gamma})]$.

Let $D_0 = 1$ and Δ_{n-k} be the determinant obtained from Δ by deleting the last k rows and columns. Then it can be easily checked that $\Delta_0, \Delta_1, \ldots, \Delta_n$ have alternating signs which establishes the maximality of P_{sys} at the Pareto optimal point.

•

Note that the Pareto optimal point given in (6) in the case of equal β_i's provides the same throughput for all users. When the β_i's are not equal the ratio of the throughputs of the users at this point is equal to the ratio of the weighting factors, indicating in this way that β_i is indeed the weight a user puts on its throughput.

Having shown the existence of a point which is Pareto optimal and which simultaneously maximizes the system power, the next question to be asked is if we can find an algorithm that starting from an arbitrary initial point could reach this optimum. Decentralizability is a desirable property of the algorithm, i.e. when updating each user should use only the throughputs of the other users and the knowledge of its own cost criterion.

Let $\gamma_1^t, \dots, \gamma_n^t$ be the throughputs observed by user k at time t. If user k tries to maximize the system power then

$$\gamma_k^{t+1} = \max_{\gamma_k} P_{\mathrm{sys}}(\gamma_1^t, \dots, \gamma_n^t) =$$

$$\max_{\gamma_k} \left\{ (\gamma_1^t)^{\beta_1} \dots (\gamma_k^t)^{\beta_k} \dots (\gamma_n^t)^{\beta_n} (\mu - \gamma_1^t - \dots - \gamma_k - \dots - \gamma_n^t)^n \right\}$$

$$= \frac{\beta_k}{\beta_k + n} (\mu - \sum_{\substack{l=1 \\ l \neq k}}^{n} \gamma_l^t)$$

Note that γ_k^{t+1} does not depend on the weighting factors of the other users and that it solely depends on the rates of the other users and its own weighting factor - a necessary property for decentralizability.

Using the above observation, we are now ready to propose a decentralized Pareto optimal flow control algorithm, that converges to a point that maximizes the product of the individual powers. According to this algorithm each user updates by using the most recent information and trying to maximize the system power.

Users are labeled $1, 2, \dots, n$ *and update in this order.*
Let a feasible $(\gamma_1^0, \dots, \gamma_n^0)$ *be any starting point.*
At step k *user* i *updates its throughput as follows*

$$\gamma_i^k = \frac{\beta_i}{\beta_i + n} (\mu - \sum_{l=1}^{i-1} \gamma_l^k - \sum_{l=i+1}^{n} \gamma_l^{k-1}). \tag{7}$$

Note the similarity of this algorithm to the greedy algorithm proposed in [3]. Both algorithms operate synchronously and use information that is available locally. The difference between the two lies in the selection of the criterion that each user optimizes at each update. In [3] each user optimized its own utility function, thus the name greedy. Here each user maximizes the system performance. Nevertheless, the structure of the two algorithms is very similar, the only difference being the substitution of $\frac{\beta_i}{\beta_i + 1}$ by $\frac{\beta_i}{\beta_i + n}$. Thus, the conclusions drawn regarding the greedy algorithm, that were presented in [10][4][5] holds for the Pareto optimal algorithm (7). We are going to present the main properties of the algorithm without proof. Proofs can be found in the cited references.

1. If algorithm (7) converges, it converges to the Pareto optimal point, that maximizes the product of individual powers.
2. Algorithm (7) is the Gauss- Seidel method for the solution of the linear system of equations (5).
3. The algorithm can be run asynchronously. Convergence of an asynchronous version of the algorithm is not guaranteed for all values of β_i. The range of β_i's for which the asynchronous algorithm converges is larger than that of the asynchronous version of the greedy algorithm because of the existence of the term $\frac{\beta_i}{\beta_i + n}$.
4. Relaxation techniques can be used to enhance the performance of the algorithm.

We are now in a position to comment on the discussion on decentralizability of the flow control power as presented in [3] by Jaffe. The requirement to have each user optimize its own cost criterion consists a noncooperative solution and leads to the Nash equilibrium. If some form of global optimality is required cooperation is essential and we have showed that we can achieve optimum solutions in a decentralizable way. The fact that the noncooperative solution is not the same as the cooperative one is well known in the game theory literature [12]. It is only under very specific situations that such an equality occurs.

VI. CONCLUSIONS

The problem of finding appropriate decentralized flow control algorithms in computer networks environment was addressed. It was shown that if the cost function of each user satisfies certain properties, an algorithm proposed in the literature could be modified so as a set of Pareto optimal points could be obtained. If power is used as the performance criterion these algorithms can not be used, but the structure of the problem can lead to simple algorithms for the M/M/1 case.

If the objective is the optimization of a global measure then the system power defined as the product of individual powers provides such a global measure. It was shown that any extremum or inflexion point of the system power is a Pareto optimal point and in the M/M/1 case it is a unique maximization point. This led to the introduction of a decentralized iterative algorithm to reach this point. Several properties of the algorithm were discussed, as well as the connection between cooperative and non cooperative solutions.

The possibility of extending the results presented to the general network case seems promising. Further studies of the convergence of the algorithm are also necessary.

REFERENCES

[1] Basar, T. and G.J. Oldser, "Dynamic Noncooperative Game Theory", *Academic Press,* 1982.

[2] Beeson, R.M., "Optimization with Respect to Multiple Criteria", *Ph.D. Dissertation,* University of Southern California, 1971.

[3] Bharath-Kumar K. and Jaffe J.M., "A new approach to performance oriented flow control", *IEEE Transactions on Communications,* Vol COM-29, April 1981, pp. 427-435.

[4] Bovopoulos, A.D. and Lazar A.A., "Decentralized Algorithms for Flow Control", *Proceedings of the 25th Allerton Conference on Communication, Control and Computing,* University of Illinois at Urbana-Champaign, October 1987.

[5] Bovopoulos, A.D. and Lazar, A.A., "Synchronous and Asynchronous Iterative Algorithms for Load Balancing", *Proceedings of the 22nd Conference on Information Sciences and Systems,* Princeton, NJ, March 1988.

[6] Cansever, D., "Decentralized algorithms for flow control in networks ", *Proceedings of the* 25th *Conference on Decision and Control, Athens, Greece, December 1986.*

[7] Chu, C.K., "On the noninferior set for the systems with vector valued objective functions", *IEEE Transactions on Automatic Control,* Vol. AC-15, October 1970, pp. 591-593.

[8] Douligeris C. and Mazumdar R., "An approach to flow control in an integrated environment", *CTR Technical Report, CUCTR-TR-50,* Columbia University, 1987.

[9] Douligeris C. and Mazumdar R., "On Pareto Optimal Flow Control in a Multiclass Environment", *Proceedings of the 25th Allerton Conference on Communication, Control and Computing,* University of Illinois at Urbana-Champaign, October 1987.

[10] Douligeris C. and Mazumdar R., "User Optimal Flow Control in an Integrated Environment", *Proceedings of the Indo-US Workshop on Signals and Systems,* Bangalore, India, Jan. 8-12, 1988.

[11] Douligeris, C. and Mazumdar, R., "A Game Theoretic Approach to Flow Control in an Integrated Environment with two Classes of Users", *Proceedings of the Computer Networking Symposium,* pp. 214-221, Washington DC, April 11-13, 1988.

[12] Dubey, P., Inefficiency of Nash Equilibria, *Mathematics of Operations Research,* Vol. 11, No. 1, 1986, pp. 1-8.

[13] Jaffe, J.M., "Flow Control Power is Nondecentralizable", *IEEE Transactions on Communications,* Vol. COM-29, No. 9, 1981, pp. 1301-1306.

[14] Sanders, B.A., "An Asynchronous, Distributed Flow Control Algorithm for Rate Allocation in Computer Networks", *IEEE Transactions on Computers,* Vol. 37, No. 7, pp. 779-787, July 1988.

[15] Yemini, Y., "Selfish Optimization in Computer Networks", *Proceedings of the* 20th *Conference on Decision and Control,* pp. 374 - 379, 1981.

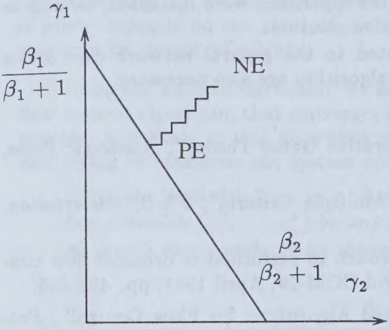

γ_1
$\frac{\beta_1}{\beta_1+1}$
NE
PE
$\frac{\beta_2}{\beta_2+1}$
γ_2

Fig. 1. Convergence of Algorithm A
(n=2)

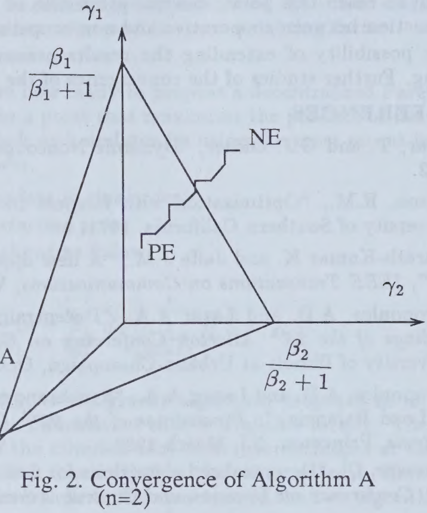

γ_1
$\frac{\beta_1}{\beta_1+1}$
NE
PE
γ_2
$\frac{\beta_2}{\beta_2+1}$
$\frac{\beta_3}{\beta_3+1}$
γ_3

Fig. 2. Convergence of Algorithm A
(n=2)

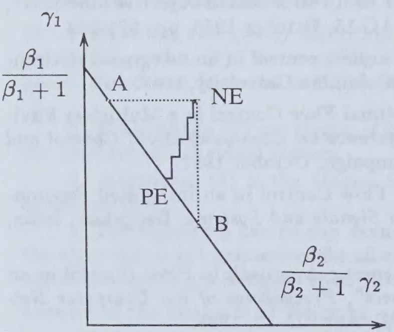

γ_1
$\frac{\beta_1}{\beta_1+1}$
A
NE
PE
B
$\frac{\beta_2}{\beta_2+1}$
γ_2

Fig. 3. Convergence of Algorithms B & C
(n=2)

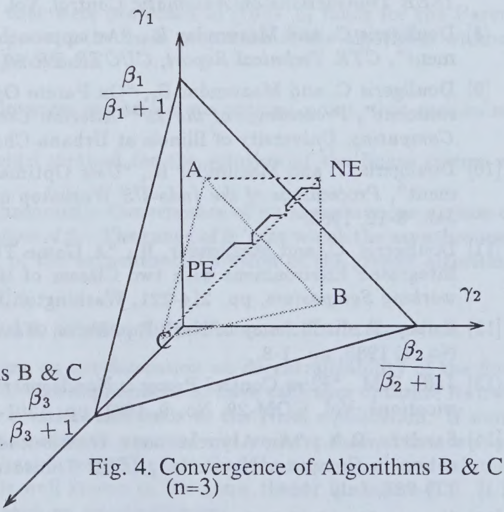

γ_1
$\frac{\beta_1}{\beta_1+1}$
A
NE
PE
B
C
γ_2
$\frac{\beta_2}{\beta_2+1}$
$\frac{\beta_3}{\beta_3+1}$
γ_3

Fig. 4. Convergence of Algorithms B & C
(n=3)

INDEX OF AUTHORS

Abed, E. H. 338
348
905
Agathoklis, P. 199
Allebach, J. 609
Anderson, B.D.O. 417
Antsaklis, P. J. 191
1160
1172
Apostolico, A. 253
Arbouz, N. M. 862
Arora, A. S. 1059
Atallah, M. J. 253
1114
Aupperle, L. J. 97
Azimi, M. 109
Baccelli, F. 965
Bai, S. 917
Ball, D. 169
Barmish, B. R. 187
Behroozi-Toosi, A. B. 979
Berger, T. 580
Bhargava, B. 368
Bhasker, J. 808
Bhattacharya, B. K. 107
Biglieri, E. 1080
Bilardi, G. 1084
Blake, I. 471
Blankenship, J. 378
Bovik, A. C. 239
Brahmadathan, K. 683
Brochin, F. M. 1001
Bross, A. 119
Browne, S. 368
Bucklew, J. A. 556
Bui, T. N. 798
Burnashev, M. V. 1059
Butler, J. T. 408
Calderbank, A. R. 1082
Callahan, K. R. 758
Cassandras, C. G. 1170
Catarci, T. 1138
Chakrabarti, C. 1015
Chakravarty, S. 740
Chandran, S. 87
Chandrasekharan, N. 283
Chang, B. 459
Chao, C. C. 1049
Chao, P. C. 1037
Chen, J. 881
Chen, L. 264
Cheng, W.-T. 750
Chern, M.-Y. 1037
Chiang, Y. W. 969
Choe, Y. 647
Chung, H. 494
Coan, B. A. 663
Cohn, D. L. 1059
Conn, H. E. 97
Costello, Jr., D. J. 1061
Coyle, E. J. 589
Crescenzi, P. 721
Dandapani, R. 758
Das, M. 629
Dasgupta, S. 229
Dattatreya, G. R. 15
Davison, E. J. 907
DeCarlo, R. 1150
Dehne, F. 1035
Delosme, J.-M. 1104
Delp, E. J. 657
Douligeris, C. 1183
Drakopoulos, E. 39
Dubois, M. 701
Erickson, C. 109
Esfandiari, F. 535
Farison, J. B. 893
Farrell, J.A. 852
Feig, E. 473
Ferreira, H. C. 1061
Forsberg, K. 471
Frazho, A. E. 872
Freudenberg, J. S. 881
Fu, J.-H. 338
Fuhrmann, D. R. 219
Fuja, T. 461
Gambosi, G. 819
Gao, Z. 191
Garber, F. D. 11
Gawronski, W. 209
Georghiades, C. N. 1
Geyer-Schulz, A. 810
Goel, D. K. 758
Golic, J. Dj. 639
Gopal, G. 991
Greenberg, A. G. 571
967
Gubner, J. A. 499

Gulati, R. K. 758
Guo, T. 356
Haddad, A. H. 937
Hall, E. B. 599
Hambrusch, S. E. 1114
Hedetniemi, S. T. 283
Heegard, C. 461
Herley, K. T. 1084
Herro, M. A. 1082
Hsu, W.-L. 842
Hu, Y. H. 1042
1046
Huang, Y.-F. 229
Hui, S. 518
Hunt, III, H. B. 788
Hwang, G.-Y. 1126
Iftar, A. 883
Ingram, M. A. 937
Italiano, G. F. 819
JaJa, J. 1015
Janko, W. H. 810
Jha, N. K. 768
Johnson, C. D. 159
Johnson, C. R. 417
Johnson, R. 274
Jou, J.-Y. 750
Juang, J.-N. 209
Jung, D.-M. 83
Kamen, E. W. 437
Kanal, L. N. 15
Kanefsky, M. 459
Kappos, E. 953
Kashyap, R.L. 647
Katsaggelos, A.K. 661
Keil, J. M. 97
Kermani, P. 566
Kesler, S. B. 457
Khalil, H.K. 535
Kinney, L. L. 778
Kolla, S. R. 893
Koppelman, D. M. 981
Krenz, G. S. 313
Kudva, P. 547
Kumar, D. 139
Kumar, P. V. 492
494
Lai, T.-H. 1094
Lakshman, T. V. 673
Larmore, L. L. 253
Lay, K.T. 661
Le, D. K. 872
Lee, C. C. 39
969
Lee, D. T. 129
Lee, P.-N. 378
Lee, S. C. 1005
Lei, T. 659
Li, J.-H. 508
Li, Y. 1140
Lin, Q. 609
Liu, C.-M. 492
Liu, L. 119
Liu, R.-W. 331
731
Liu, Z. 965
Logan, D. 119
Loh, N. K. 629
Longman, R. W. 927
Lou, R. D. 1136
Lu, H. 1005
Madani-Esfahani, S. M. 518
Madisetti, V. K. 149
Malladi, S. S. 1061
Manacher, G. K. 832
Mankus, T. A. 832
Mareels, I. M. Y. 417
Marinescu, D. C. 955
Massey, W. A. 967
Mazumdar, R. 1183
McClamroch, N. H. 356
537
McEliece, R. J. 1049
McFaddin, H. S. 253
McLane, P. J. 1080
Melhem, R. 1126
Messerschmitt, D. G. 149
Metzner, J. J. 482
Michel, A. N. 508
852
Milenkovic, P. H. 1042
Miller, D. E. 907
Miller, M. I. 1040
Miller, R. K. 313
Milosavljevic, M. M. 251
Moitra, A. 274
Moitra, D. 303
Moulin, P. 27
Mount, D. M. 87
Murthy, G. R. 589
Naganathan, S. 1046
Nakajima, K. 398
Nandy, S. K. 1025
Narasimhamurthi, N. 356
Nayeri, M. 447
Ney, P. 556
O'Rourke, J. 49

97
O'Sullivan, J. A. 27
Oloomi, H. 903
Oruc, A. Y. 981
Ozguner, U. 883
Panwar, R. B. 1025
1116
Passino, K. M. 1160
1172
Patnaik, L. M. 1116
Paulik, M. J. 629
Pawlak, M. 366
Peck, A. 798
Peczkowski, J. L. 177
Peir, J.-K. 119
Peleties, P. 1150
Pham, Q. T. 1035
Phan, M. 927
Porod, W. 508
Prince, J. L. 619
Ramakrishnan, T. 778
Raman, V. 331
Ramanan, P. 58
Rao, A. K. 229
Rao, R. R. 979
Ravi, S. S. 388
788
Rodger, S. H. 293
Rosberg, Z. 566
Rosen, Y. 408
Rosenkrantz, D. J. 388
Roysam, B. 1040
Sadowsky, J. S. 556
Sain, M. K. 177
189
Salam, F. M. A. 917
Sallam, B. M. 457
Salowe, J. S. 73
Samad, T. 808
Sanchis, L. A. 711
Sandberg, I. W. 169
Sanders, K. D. 239
Sarrafzadeh, M. 1136
Sartori, M. A. 1172
Sastry, C. R. 437
Sawan, M. E. 903
Saydy, L. 348
Schevon, C. 49
Schrader, C. B. 189
Schulz, T. J. 13
Sennott, L. I. 569
Sethares, W.A. 417
Sewchand, W. 659
Shen, D. 119
Sherman, D. L. 249
Shi, Z. 187
Shih, F.-H. W. 398
Silvestri, R. 721
Simaan, M. 437
939
Sira-Ramirez, H. 321
Skavantzos, A. 1044
Snorrason, O. 11
Snyder, D. L. 13
27
Sreeram, V. 199
Srivastava, P. 25
Stark, W. E. 1070
Stearns, R. E. 788
Stojmenovic, I. 1035
Stouraitis, T. 1044
Strickland, S. G. 1170
Su, R. 862
Sugihara, K. 63
Suzuki, I. 63
Syrmos, G. L. 905
Taipale, D. 480
Talamo, M. 819
Tamboli, A. 378
Tan, H. L. 657
Taudes, A. 810
Telang, V. 1082
TeWinkel, L. E. 1114
Thomas, J. B. 1001
Tits, A. L. 348
Tong, L. 731
Tsai, K. 842
Uresin, A. 701
Vanderbei, R. J. 571
Vastola, K. S. 37
Venkatesan, S. 683
Vidyasankar, K. 693
Wang, C. A. 85
Wang, D. 537
Warren, D. J. 35
Wei, V. K. 673
Weinrib, A. 991
Wessel, A. E. 599
White, W. 1094
Willett, P. 25
Willsky, A. S. 619
Wise, G. L. 599
Woerner, B. D. 1070
Wolf, J. K. 1048
Wong, Y. 1104
Wonham, W. M. 1140

Wu, Q. 219
Wyman, B. F. 177
Yalamanchili, S. 129
Yamashita, M. 63
Yang, J. C. S. 547
Yang, T.-C. 547
Yanushevsky, R. T. 501
Yaz, E. 948
Ye, Z. 580
Yesha, Y. 264
Yin, G. 427
Young, K. D. 528
Yousef, H. 939
Zak, S. H. 518
Zazanis, M. A. 549
Zhang, L. 731
Zhu, J. 159
Zoltowski, M. D. 249

Wu, Q. [illegible]
Wyman, B. F. 171
[illegible], S. 220
[illegible], M. 89
Yang, J. C. S. 547
Yang, Y. C. 567
[illegible], A. T. 597
Yeo, P. 498
Yu, J. 430
[illegible] 599
Wu, C. 433
Young, R. D. 549
[illegible], H. 859
[illegible], S. S. 918
[illegible] 588
[illegible] 771
[illegible] 189
[illegible] 249

UNIVERSITY OF ILLINOIS-URBANA
621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1988:2
3 0112 008149251